SYBASE®
An SAP® Company

Mobile Application Life Cycle

Sybase Unwired Platform 2.2 SP02

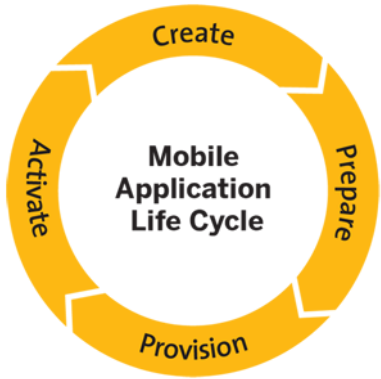
DOCUMENT ID: DC01855-01-0222-03
LAST REVISED: April 2013

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

Contents

Mobile Application Life Cycle

Sybase® defines the mobile application life cycle as the process by which an enterprise gives users access to applications and other related mobility resources (enterprise data, network connections, and so on), thereby allowing the application to mediate the user interaction with your organization in a sustained and controllable fashion.



Stage 1: Create

The create stage consists of package and application development for online, offline, and hybrid applications using a corresponding development archetype.

The correlation of application to package varies depending on the application type:

- For offline applications, one or more MBO packages can be assigned to an application. MBO packages that are used in more than one application must be assigned to each application when the application is defined in Sybase Control Center.
- For Hybrid App applications, all MBO packages are accessible in all domains. You need not assign MBO packages if you are using Hybrid App, which is the default Hybrid App application (HWC). However, for a customized container with an application ID other than the default, the corresponding application's MBO packages must be selected when the application is defined and paired with its application ID.
- OData SDK Application require no MBO assignments; packages are not used by this application type.
- REST API applications do not use MBO packages . The application can be developed with any language and any platform that supports HTTP request/response.

Packages

For Object API applications and Hybrid Apps, a package is a named container for one or more mobile business objects (MBOs).

Object API and Hybrid Apps can be associated with multiple packages that extract data from multiple data sources.

For an introduction to different mobile data models and development archetypes, see *Mobile Data Models* and *Mobile Application Development* in *Fundamentals*

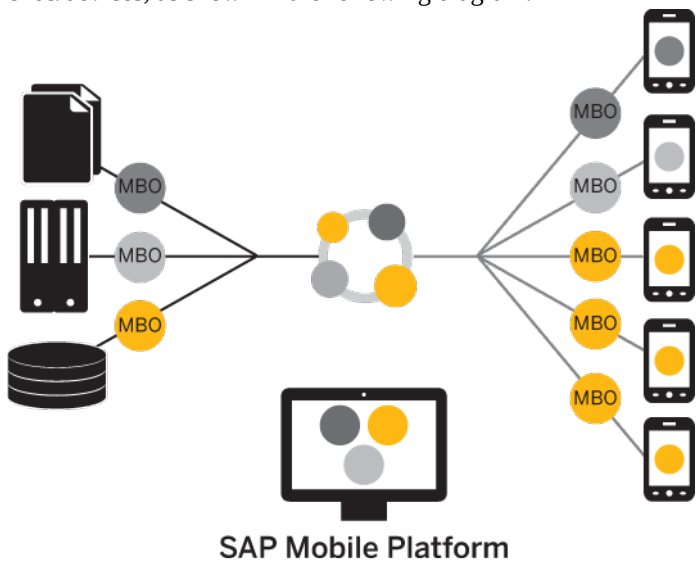
Mobile Business Object Overview

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 Unwired Server, and accessed from mobile device application client code generated from Sybase Unwired WorkSpace or by using command line tools. MBOs:

- Are created using the Sybase Unwired 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 Unwired 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 Unwired Server during package deployment.

When the data model is complete, code artifacts are generated. The MBO package, containing one or more MBOs is deployed to Unwired Server. Other MBO artifacts are used to develop a mobile application using Native Object API or 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.

Best Practices for Moving Packages to Production Servers

When moving from a preproduction environment to a production environment, there are several best practices to keep in mind.

Sybase recommends that you:

- **Connections –** Remember to update domain-specific connections for packages assigned to that domain. Preproduction servers vary from production servers used in a live environment. The connection profile used by an MBO must use correct values to production servers. See *Editing Connection Profile Properties* in *Sybase Unwired WorkSpace - Mobile Business Object Development*. If you are pooling connections, you will need to use an appropriate production domain connection template for these packages. See *Creating Connections and Connection Templates* in *Sybase Control Center for Sybase Unwired Platform*
- **Domains –** Ensure production-ready applications register connections to the same domain as the production-ready package. If there is a mismatch between a registered connection and a deployed package, the application cannot find corresponding packages. Validate that application connections and packages match. If they do not, either reregister the application connection using the correct domain assignment, or redeploy the package to the correct domain.

Applications

An application consists of two parts: the software user interface that consumes enterprise data from one or more data sources, and the definition created in Sybase Control Center that allows the mobile application to be recognized by the runtime. Depending on the application type, the application definition on the server can establish the relationship among packages used in the application, the domain that the application is deployed to, the activation code for the application, and other application specific settings.

Unwired Platform supports multiple options for development and deployment of mobile applications.

- **Object API applications –** Object API applications are customized, full-featured mobile applications that use mobile data model packages, either using mobile business objects (MBOs) or Data Orchestration Engine, to facilitate connection with a variety of enterprise systems and leverage synchronization to support offline capabilities
- **Hybrid Apps –** HTML5/JS Hybrid Apps support simple business processes, such as approvals and requests, and also use mobile business objects (MBOs) to facilitate connection with a variety of enterprise systems. With this approach, a hybrid web container is developed and deployed to a device, then one or more workflows are deployed to the container.
- **OData applications –** OData applications are simple, mostly online mobile business applications that leverage standards-based approaches—OData protocol and RESTful Web Services.
- **Agentry applications –** Agentry applications provide the ability to support one or more client device types and one or more enterprise information systems of varying types from a single application project, high levels of configuration and modification with minimal user impact, simple or complex multi-screen workflows, and high data volumes stored on client devices.
- **REST API applications –** Applications developed using 3rd Party SDKs, such as Sencha and PhoneGap, for any device platform, and using any programming language that supports sending and receiving HTTP messages can use the Sybase Unwired Platform REST APIs to call services provided by the Sybase Unwired Platform Runtime.

While developers create the client application using either Sybase Mobile SDK or a third-party SDK, applications must be defined, then managed and monitored by administrators in Sybase Control Center. The stages in the life cycle of an application may be different for different application types.

Recommended Test Methodologies

There are several different test methodologies you can use, each having their own goals, procedures, and tools. Be prepared to perform functional, load, and performance testing on the applications you create before distributing Sybase Unwired Platform applications to end users.

- **Functional Testing –** The goal of functional testing is to ensure the application or program works according to the user requirements, and typically include unit testing, integration testing, and end-user testing. Recommended tools:
  - Testing and debugging tools or the IDE, frequently on a simulator during the development phase.
  - SAP Solution Manager to manage test cases.
- **Load Testing –** The goal of load testing is to see if the application or system will fail under load. It is similar to performance testing in that you want to see how the system performs,

and it is similar to functional testing in that you are looking to see if there are bugs or poor implementation aspects that cause the system to fail at certain loads. Recommended tools:

  - Recording tools for generating test scripts.
  - Parsing tool to parameterize test script.
  - Test framework that simulates load and collects results.
- **Performance Testing –** The goal of performance testing is to measure the system throughput while under load. Performance testing has the same requirements as load testing, only the focus is on measurement and analysis. The performance criteria that are measured during performance testing are called metrics. Recommended tools:

  - Agents to measure performance metrics.
  - Framework to manage results.
  - Analysis tools to interpret results and identify bottlenecks and points of failure.

For more information on how to perform these tests using Sybase and third-party tools, see *How To ... Test the Sybase® Unwired Platform: Available Test Tools* on Sybase Community Network at *http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/c03d916b-945f-2f10-ab9c-af72a97fc0f7?QuickLink=index&overridelayout=true&54395760825565* .

Best Practices for Testing Applications on a Physical Device

All applications should rigorous functional, load, and performance testing on a physical device before deploying them to your end users. Because there may be differences between development, test, and production environments, always use a physical device to perform these tests over the same network type that your end users will use.

In some cases, applications that work on a simulator or emulator may fail once installed onto a device. The testing applications pro-actively exposes configuration or design issues early on. Sybase recommends that you watch for these potential issues:

Table 1. When Moving from a Simulator to a Device

Practice to Consider	Object API Application	REST API applications	HTML5/JS Hybrid Apps	OData SDK Application	Agentry applications
Ensure the server node is available from outside the firewall. Simulators run on an internal network, so when moving to a WiFi or wireless network, connectivity occurs through a proxy or a Relay Server. Ensure you configure RSOE properties in Sybase Control Center accordingly. See *Configuring RSOE Connection Settings* in *Sybase Control Center for Sybase Unwired Platform*.	Yes	Yes	Yes	Yes	Yes
Ensure the application uses names the correct domain for security configurations and/or packages. Frequently, developers specify a different domain for testing environment.	Yes	Yes	Yes.	Yes	
When connecting to an EIS, ensure the credentials required by the endpoint are correct for the environment you are connecting into. If you have preproduction instance, these credentials may not be the same.	Yes. Credentials are sent in the HTTP request.	Yes. Credentials are configured in the proxy connection pool.	Yes.	Yes.	
For networks that use an HTTP proxy server, ensure connection pool used by the application has the *EnableHttpProxy* set to `true`. See *Creating Connections and Connection Templates* in *Sybase Control Center for Sybase Unwired Platform*.		Yes			

Practice to Consider	Object API Application	REST API applications	HTML5/JS Hybrid Apps	OData SDK Application	Agentry applications
Add the corresponding proxy server values in the User Options for the Unwired Server profile. For example: `-Dhttp.proxyPort=8080 -Dhttp.proxyHost=`*`proxyName`* See *Configuring SAP Mobile Server to Securely Communicate With an HTTP Proxy* in *Sybase Control Center for Sybase Unwired Platform*.		Yes			
For BlackBerry devices the application also needs to be signed using the BlackBerry Signing Authority Tool, in addition to using the Blackberry code signing API for sensitive applications. For details, see *http://supportforums.blackberry.com/t5/Java-Development/Protect-persistent-objects-from-access-by-unauthorized/ta-p/524282*. For an example of how code signing is used, see *Migrating BlackBerry Applications* in *Developer Guide: Migrating to Sybase Mobile SDK*.	Yes				
For iOS devices, monitor memory usage once the application is installed on the device, particularly on iPhones. Since the memory is limited on iPhone devices, define and release local autorelease pools frequently. Frequent releasing allows the auto-released objects to be promptly flushed from memory.	Yes				

Practice to Consider	Object API Application	REST API applications	HTML5/JS Hybrid Apps	OData SDK Application	Agentry applications
For iOS devices, monitor memory usage once the application is installed on the device. Because memory is limited on iOS devices, define and frequently release the local auto-release pool. Frequent releasing allows the auto-released objects to be promptly flushed from memory. See *Managing the Background State* section in *Developer Guide: iOS Object API Applications*.	Yes				
For iOS applications, ensure the application is adjusted to use the APNS API to receive push notifications. The APNS API can only be used on applications installed to a device.	Yes			Yes	
Deploy network edge certificates signed by a real CA. HTTP connections, or self-signed certificates will not work once you try connecting from outside a firewall.	Yes	Yes	Yes	Yes	

Stage 2: Prepare

In the prepare stage you are preparing Unwired Server to support mobile applications and serve data to them.

For Unwired Server to support mobile applications requires that the server be prepared by configuring the server and deploying packages in a pre-production Unwired Platform environment. Once the pre-production runtime environment is stable and package and application development complete, deploy the packages to the production environment hosts.

Note: Agentry applications have different steps in the prepare stage. For more information, see *Creating Agentry Application Definitions* in *Sybase Control Center for Sybase Unwired Platform*.

Configuring Native Notification Environments

If the application is using native push notifications, third-party environments need to be setup to channel notifications from Unwired Server. Then, when the administrator enables native push in the application connection template, push notifications from Unwired Server are delivered seamlessly

APNS Environments

Apple Push Notification Service (APNS) delivers user notifications on iOS devices. To use APNS for push notification, you must enable APNS notifications for Unwired Server.

iOS devices do not require a runtime or a client; only the application must be deployed to the device. The lack of a runtime allows users to self-manage their devices: device users download application files to their iOS devices and synchronize updates as required.

Apple Push Notification Service (APNS) allows users to receive notifications. APNS:

- Must be set up and configured by an administrator on the server.
- Must be enabled by the user on the device.
- Can be used with any device that supports APNS. Some older Apple devices may not support APNS.
- Cannot be used on a simulator.

Configuring Apple Push Notification Service

Use Apple Push Notification Service (APNS) to push notifications from Unwired Server to the iOS application. Notifications might include badges, sounds, or custom text alerts. Device users can use Settings to customize which notifications to receive or ignore.

Prerequisites

Perform these prerequisites in the Apple Developer Connection Portal:

- Register for the iPhone Developer Program as an enterprise developer to access the Developer Connection portal and get the certificate required to sign applications.
- Create an App ID and ensure that it is configured to use Apple Push Notification Service (APNS).
- Create and download an enterprise APNS certificate that uses Keychain Access in the Mac OS. The information in the certificate request must use a different common name than the development certificate development teams might already have. This is because the enterprise certificate also creates a private key, which must be distinct from the development key. This certificate must also be imported as a login keychain and not a system keychain and the developer should validate that the certificate is associated with the key in the Keychain Access application. Get a copy of this certificate.

  Note: A new 2048-bit Entrust certificate needed for APNS.

  Apple uses a 2048-bit root certificate from Entrust, which provides a more secure connection between Unwired Server and APNS. This certificate comes with the Windows OS, and is upgraded automatically with Windows Update, if it is enabled. This information is not part of the procedure that documents APNS support.

  If Windows Update is disabled, you must manually download and install the certificate (entrust_2048_ca.cer). Go to *https://www.entrust.net/downloads/root_index.cfm*. For help on installing the certificate, see *http://www.entrust.net/knowledge-base/technote.cfm?tn=8282*.
- Create an enterprise provisioning profile and include the required device IDs with the enterprise certificate. The provisioning profile authorizes devices to use applications you have signed.
- Create the Xcode project ensuring the bundle identifier corresponds to the bundle identifier in the specified App ID. Ensure you are informed of the "Product Name" used in this project.
- Use the APNS initialization code.

Developers can review complete details in the *iPhone OS Enterprise Deployment Guide* at *http://manuals.info.apple.com/en_US/Enterprise_Deployment_Guide.pdf*.

Task

Each application that supports Apple Push Notifications must be listed in Sybase Control Center with its certificate and application name. You must perform this task for each application.

1. Confirm that the IT department has opened ports 2195 and 2196, by executing:

   ```
   telnet gateway.push.apple.com 2195
   telnet feedback.push.apple.com 2196
   ```

   If the ports are open, you can connect to the Apple push gateway and receive feedback from it.
2. Upload the APNS certificate to Sybase Control Center:
   a) In the navigation pane, click **Applications**.
   b) In the administration pane, click the **Applications** tab.
   c) Select the application for which you want to enable APNS, and click **Properties**.
   d) Click the **Push Configurations** tab and click on **Add**.
   e) Configure all required properties, including the corresponding password and upload the certificate. See *APNS Native Notification Properties* in *Sybase Control Center for Sybase Unwired Platform* online help.
3. Deploy the iOS application with an enterprise distribution provisioning profile to users' iOS devices.
4. Verify that the APNS-enabled iOS device is set up correctly:
   a) In Sybase Control Center, ensure the user has already activated the application and is connected to the Unwired Server, by looking for the corresponding entry in **ApplicationsApplication Connections**.
   b) Validate that in the Application Connection ID, the application name appears correctly at the end of the string.
   c) Select the user and click **Properties**.
   d) Check that the *APNS Device Token* contains a value. This indicates that a token has passed successfully following a successful application activation
5. Verify that native notification is enabled for the user:
   a) Select the user name and click **Properties**.
      - For Application Settings, ensure the **Notification Mode** property is set to either **Only native notifications** or **Online/ payload push with native notification**.
      - For Apple Push Notifications, ensure the **Enabled** property is set to **True**.
6. Test the environment by initiating an action that results in a new message being sent to the client.

   If you have verified that both device and server can establish a connection to the APNS gateway, the device receives notifications and messages from the Unwired Server. If you configured **Online/ payload push with native notification**, allow a few minutes for the delivery. If the device is offline and the message is pending in messaging queue, Unwired Server triggers the native push notification mechanism to send the Pending Items to the

device via APNS. See *Reviewing the Pending Items Count for Messaging Applications* in *System Administration*.

Note: Notifications require a connection to APNS on port 5223. This port is not always routed through the firewall on corporate wireless networks.

7. To troubleshoot APNS, use the *SUP_HOME*\Servers\Unwired Server\logs \server log file.

BES Environments

BlackBerry client applications can connect to a production environment using Relay Server and use BlackBerry Enterprise Server (BES) to provision applications and application-related artifacts.

BlackBerry application connections can occur over a cellular or Wi-Fi network However, Wi-Fi is only available when enabled on BES.

BES Components Required By Unwired Platform

BlackBerry Enterprise Server (BES) is the middleware software package that is part of the BlackBerry wireless platform from Research In Motion®. BES software and services connect to Unwired Platform on the corporate LAN and redirect data sent by Unwired Server.

BES uses Windows services to carry out the basic operations of the system. While many services exist, these are the ones you can use with Unwired Platform:

- **BlackBerry Messaging Agent –** Performs wireless calendar synchronization, generate initial encryption keys, or provide email and lookup services.
- **BlackBerry MDS Connection Service –** Pushes requests from intranet applications. The BlackBerry MDS Services (for Apache Tomcat) sends and receives Web browsing to the device through a BlackBerry Dispatcher service. You can use these services for Unwired Platform data flow with third-party Java applications on-device.
- **BlackBerry Policy Service –** Pushes wireless IT policies to devices, sets commands for device locks and remote wipe, and generates new encryption keys.
- **BlackBerry Router –** Routes all Unwired Platform data to wireless devices by establishing a link between BES (which can be installed on the same host as the router) and SRP host.
- **BlackBerry Synchronization Service –** Performs OTA backup and synchronization of all PIM data (contacts, tasks and notes), except calendar, which is instead performed by BlackBerry Messaging Agent.

For information about any of these or other services, see the BES documentation at *http://docs.blackberry.com/en/* .

BES Requirements

Review requirements for using Sybase Unwired Platform in a BES environment.

Unwired Platform Installation and Configuration

Recommendation	Where to Find Information
(Optional) Install a Relay Server if Unwired Server is behind the internal firewall, then configure it prior to provisioning the application to the device.	*Installation Guide for Runtime*

BlackBerry Enterprise Server Requirements

Prerequisite	Where to Find Information
Configure push notifications (if required) from Unwired Server to each BES.	*Configuring BlackBerry Push Settings* in *Sybase Control Center online help*
Updating BES configuration for use with Unwired Platform	*Updating BES Configuration for Unwired Platform Integration*
Ensure BES can manage Unwired Platform push notifications as push requests.	*Managing Unwired Platform Push Notifications on BES*
Add Unwired Platform device users to each BES.	*BlackBerry Enterprise Server Administration Guide*
Generate an activation password, then send account information (e-mail address and password) to device users.	*BlackBerry Enterprise Server Administration Guide*
Ensure BES, hosted or unhosted, has access to your internal network.	Contact the system administrator for your internal network.

Preinstallation Device Configuration

Have device users complete the device preinstallation tasks before provisioning the application to the device.

Prerequisite	Where to Find Information
Install device prerequisites. The Mobile Sales installation for BlackBerry smartphone requires an SD card with at least 100MB free storage space. The Mobile Sales database and log file are stored on the SD card.	*Sybase Mobile Sales for SAP CRM Device Users Guide for BlackBerry > Installation Prerequisites* See *Developer Guide: BlackBerry Object API Applications*, and *Provisioning Security Artifacts* in *Security*.
Provide enterprise activation information so device users can pair their devices to the BlackBerry Enterprise Server.	*http://na.blackberry.com/eng/support/enterpriseactivation/*

Prerequisite	Where to Find Information
(Optional) If users need it to self-install the Unwired Platform application, install BlackBerry Desktop Software on a personal computer, which includes Desktop Manager. Sybase only recommends this for smaller organization that manage a small number of devices.	*http://na.blackberry.com/eng/services/desktop/*

Updating BES Configuration for Unwired Platform Integration

For Unwired Platform to connect with BES, you must change BES properties so that communication with Unwired Server occurs.

Note: Beyond the BES, there are no Unwired Platform-specific changes required for MDS Connection services or the device simulator. An application-reliable port setting should not be needed.

1. Change the URL of the BES server and the port (in the *rimpublic.property*) to `WebServer.listen.port=8080`.
2. If you require push notifications to the BlackBerry device, enable and set:
   `push.application.reliable.ports=102`
3. Increase the BES limit on response size to one more appropriate for Unwired Platform usage.

   By default, the BlackBerry Enterprise Server (BES) limits the response size of a single HTTP response to 128K. If you try to fetch anything bigger, your application might receive a `413 (Request Entity Too Large)` response code.

   a) Open the BES management console.
   b) Change the **Maximum number of kilobytes per connection** to a higher value, up to 1024K.

Configuring BlackBerry Push Settings for Unwired Server

Create a new BlackBerry Push Service configuration that specifies the delivery method for service users. Typically, push notifications are routed to the MDS connection service as push requests.

BlackBerry push notifications alert offline users to the availability of new items awaiting retrieval on Unwired Server. Push notification uses an IP connection only long enough for the Send/Receive data exchange to complete. BlackBerry Push notifications overcome issues with always-on connectivity and battery life consumption over wireless networks. Configuring push notifications is just one part of the BlackBerry environment setup.

1. Obtain the MDS connection service URL.

a) Open the BlackBerry Manager, then from the Explorer View, click **BlackBerry Solution topology > BlackBerry Domain > Servers > *ServerName*** .
b) Click **MDS Services**, and copy the MDS Connection Services Server URL.

2. In Sybase Control Center, or Application Settings, ensure the **Notification Mode** is set to either **Only native notifications** or **Online/ payload push with native notification**, using the MDS service URL you recorded.

   For details, see *Configuring Native Notifications* in *Sybase Control Center for Sybase Unwired Platform*.

   Note: If you configured **Online/ payload push with native notification**, allow a few minutes for the delivery or notification mechanism to take effect and before the value in Pending Items in Sybase Control Center changes accordingly. See *Reviewing the Pending Items Count for Messaging Applications* in *System Administration*.

Managing Unwired Platform Push Notifications on BES

The BlackBerry MDS Connection Service receives push application requests from Unwired Server via push notifications you configure in Sybase Control Center. Unwired Server sends the notifications on to MDS as push request that are then delivered to Unwired Platform device applications on BlackBerry devices.

You can control how the BlackBerry MDS Connection Service processes, stores, and sends push notifications as device application requests. For more information about types of push requests, visit *http://www.blackberry.com/developers* and review content in the *BlackBerry Java Development Environment Development Guide*.

Preparing the MDS Connection Service for Unwired Server Push Notifications

Configure multiple options for the MDS Connection Service from the BlackBerry Administration Service.

1. Open the BlackBerry Administration Service, then from the Servers and components menu, click **BlackBerry Solution topology > BlackBerry Domain > Component view > BlackBerry MDS Connection Service**.
2. Choose the service instance and click **Edit**.
3. Configure any of the following:

Service Property	Description	Configuration Details
Device ports	When a BlackBerry application receives an application-reliable push request, it sends a delivery confirmation message to the MDS Connection Service. This message is then forwarded to Unwired Server.	1. Contact your application developers for the unique port numbers used in the BlackBerry applications that support application-reliable push requests. 2. In **Device ports enabled for reliable pushes**, enter the device port. 3. Use **Add** to add multiple ports.
Storing push requests in the BlackBerry configuration database.	To manage memory and system resources in your organization's environment, configure the MDS Connection Service to store Unwired Server, Push Access Protocol (PAP), and other Research In Motion push connections in the Configuration Database. You can also configure storage settings for the Configuration Database. For more information about types of push requests, see *http://www.blackberry.com/developers* and review the contents of *BlackBerry Java Development Environment Development Guide*.	For PAP, in **Maximum number of push messages stored**, enter a number that reflects the total number of push notifications you want the Configuration Database to store.
Configuring the maximum number of active push connections	You can configure the maximum number of push connections that the MDS connection service can process simultaneously. If active connections exceed this value, the MDS connection service queues these push connections.	For Push access protocol, in **Maximum number of active connections**, type a number that reflects the number of push notifications required.
Configuring the maximum number of queued push connections	The MDS Connection Service queues push requests when the number of connections exceeds a limit that you specify for active connections. When the number of queued requests exceeds this limit, the MDS connection service sends a `service unavailable` message to BlackBerry devices.	For Push access protocol, in **Maximum number of queued connections**, type a number that reflects the number of push notifications required.

4. Save all changes and restart the service instance.

Deleting Unwired Server Push Requests from the Connection Queue

An automated process runs daily to delete outstanding requests from the push request queue on supported databases. However, you can also delete requests manually on an ad hoc basis as required.

1. Perform one of these actions:
   - If you are using the Microsoft SQL Server Enterprise Manager, navigate to Console Root`\Microsoft SQL Servers\SQL Server Group \<BlackBerry_Configuration_Database_server>\Management \SQL Server Agent\Jobs`.
   - If you are using the Microsoft SQL Server Management Studio, navigate to SQL Server `Agent\Jobs`.
   - If you are using an IBM DB2 UDB server, create a custom job to purge IBM DB2 UDB push requests from the Configuration Database. For simplicity, you can name the job after SQL Server processes. See step 2.
2. Start the `RIMPurgeMDSMsg<database_name>` process.

Pairing the BlackBerry Client Device to a BES Server

Pair BlackBerry devices with BES server.

Pairing devices allows you to send and receive device email (as opposed to Web mail). This task is optional for all applications except mobile workflows.
To pair devices with servers, contact the BES server administrator.

Setting JavaScript Policy

IT departments might set a specific JavaScript policy for BlackBerry devices.

If you are supporting a Hybrid App, the policy chosen does not adversely affect Hybrid Apps.

1. Determine the policy needed.
2. Enable or disable JavaScript in the built-in browser application as required.

GCM Environments

Google Cloud Messaging (GCM) is a free service for sending messages to Android devices. GCM requires an API Key to allow Unwired Server to send push notifications over GCM

Review GCM documentation on the Android developer's site, then enable GCM push when you create an application connection.

Configuring the Application to Use an GCM API Key

Enable GCM for the application connection by obtaining an API Key and enabling GCM with it.

1. Go to the Android Developer website and obtain the necessary API key.
2. In Sybase Control Center, add a GCM notification in the **Push Configuration** for the application.
   For details, see *Android Push Notification Properties* in *Sybase Control Center for Sybase Unwired Platform* .
3. Use the generated key to configure **API key**.
4. Once the application has been registered and activated, check the connection for the application and ensure that:
   - **Enabled** is set to `true`.
   - **Registration ID** and **Sender ID** are set to a value you'd expect.

Deploying Application Packages

Object API or Hybrid applications require one or more packages. Packages are objects that control the data synchronization logic for an application. Deploying is the process whereby whole or part of an application package is loaded onto an Unwired Server as one or more deployment units. Developers can deploy these packages to development environments, however administrators will control the deployment process for pre-production and production environments.

Sybase recommends that you deploy packages to a pre-production environment to validate the implementation and configuration before deploying to production hosts.

Deployment Strategies for Production Environments

When deploying multiple packages that are used by one or more applications, consider whether to use a shared, dependent, or independent deployment strategy.

A deployment strategy is typically determined by the availability, capacity, and versioning requirements of the applications that access the package logic in order to consume enterprise data.

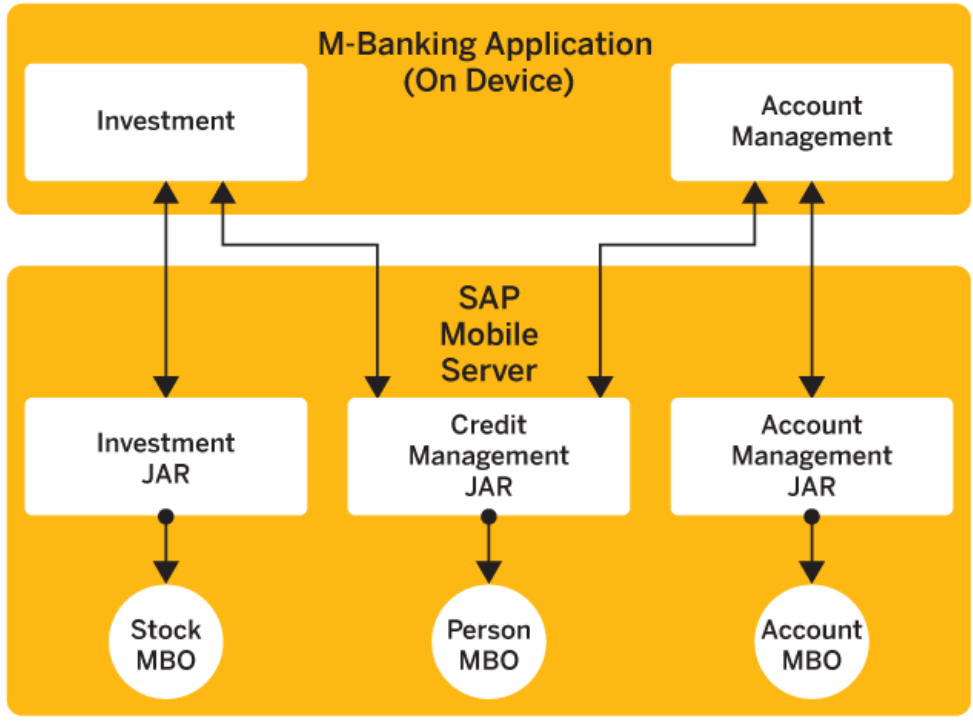
Shared Deployment Strategy

In a shared deployment approach for packages belonging to a native development model, consider an application for mobile banking that relies on different packages to manage data relating to investment management, account management, and that offers a combined view for credit management. One deployment strategy is to deploy all packages as one mobile banking unit.

For example, being able to review investment data is a frequently used feature of mobile banking, the Investment packages allows the application user to check the changing status of a

stock portfolio. Whereas, the Account Management packages might be used by the M-Banking application more sporadically. Both Investment and Account Management depend on Credit Management packages for data management, but because both have unique capacity and availability requirements, sharing packages is less of a concern.

However, the key limitation with shared deployment is versioning relating to changes or major enhancements to either the application or the packages it relies on. In this strategy all packages and all applications must be maintained simultaneously.

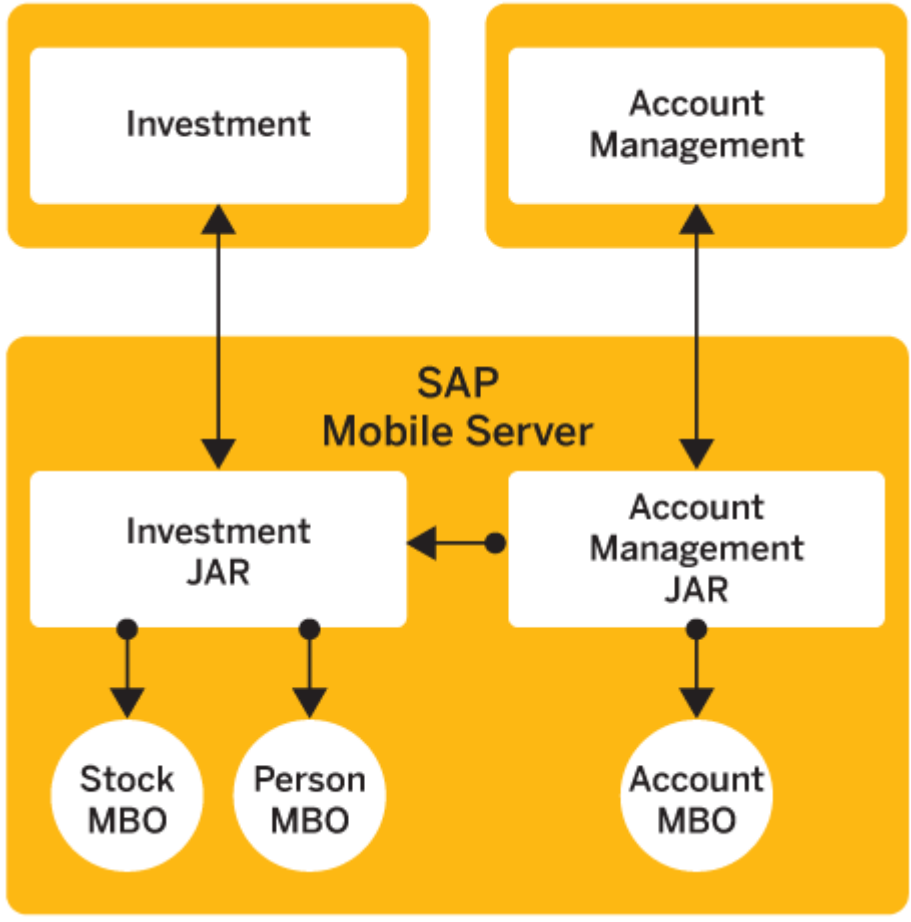


Dependent Deployment Strategy

With a dependent deployment strategy, some packages are published individually, as standalone units, while others are combined and published as multipackage units. In addition, rather than have a single M-Banking application, each feature is provisioned and installed separately.

In this use case, the goal is to keep the Investment functionality independent and self-contained and so the development and enhancement roadmap can continue without any external dependencies. This separation is desirable, since the availability requirements are most stringent for the Investment application. Conversely, the Account Management application is developed and designed to still depend on the MBOs in the Investment package, without sharing the package directly. For this strategy, the Account Management application

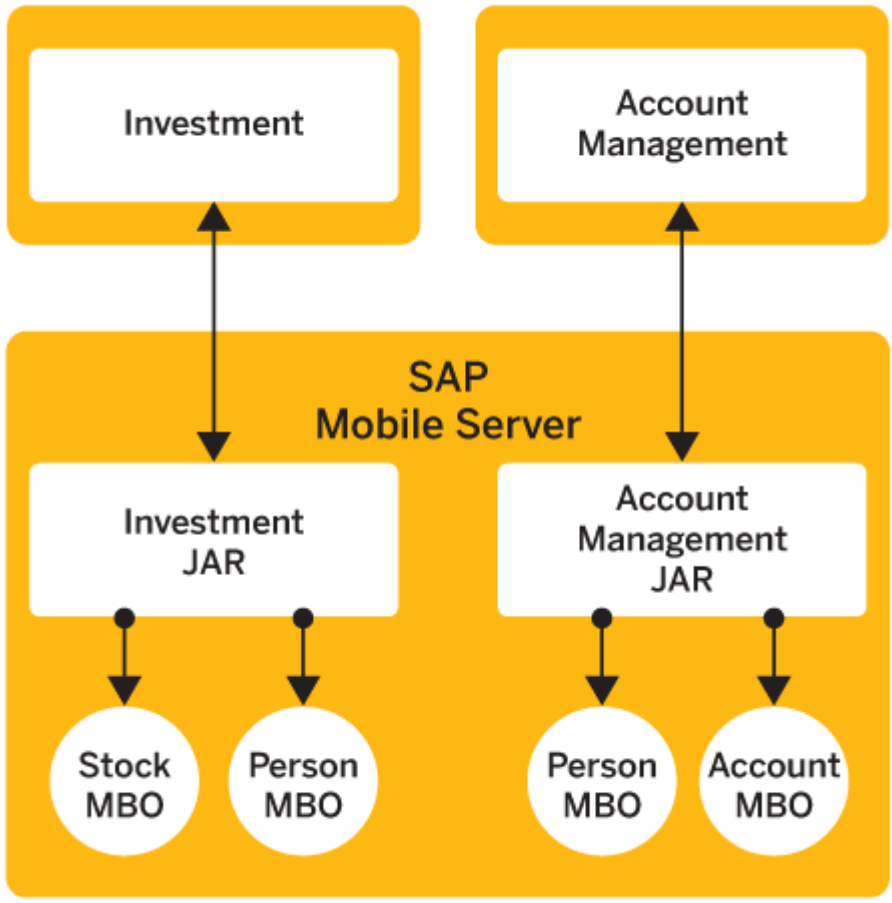
now subscribes to MBO data in the Investment package, and is instead assigned only to the Account Management package.



Independent Deployment Strategy

In independent deployment, the applications are still provisioned and installed on the device separately, and the packages they are assigned to are deployed separately by including all MBOs used by the application in a single package. There are no external dependencies in this strategy.

To make both applications and required packages for each self-contained the common Person MBO has been included in each package. That way, if the Investment application requires an upgrade, the enhancement can continue without adversely affecting the functionality of the Account Management application: the Person MBO can remain at the previous version until such time an upgrade is required.



Deployment Methods Use Cases

In development or testing environments, members of these teams can deploy directly to development instances of Unwired Server by using a supported deployment method. However, in production environments, development teams are typically restricted from performing these direct deployment activities; instead, the generally accepted practice is to package their deployment units in required format (for example, a .ZIP file for Hybrid App packages), then pass the deployment binary to the corresponding domain administrator for deployment.

Table 2. Deployment Comparison Table

Method	System Life Cycle Stage	Volume	Restrictions
`supadmin.bat`	Automates the deployment to pre-production or production Unwired Servers. This utility is typically used within a custom-developed deployment script or for CTS.	Large	Typically not used by development teams, or for the deployment of isolated packages.
Deploy wizard in Sybase Control Center	Administrator managed to pre-production or production Unwired Servers.	Small or individual	None.
Package export and import from Sybase Control Center	Administrator-controlled bulk transfers between development and pre-production environment, since the data and Unwired Server or cluster configuration is not as sensitive to differences. See *Transporting Packages Between Environments with Export and Import* in *Sybase Control Center for Sybase Unwired Platform*	Medium to large	Transferring package between development and production environments, since the backend data and Unwired Server or cluster configuration are different.
Package transport with CTS	Transferring the complete set of deployed packages in SAP environments among pre-production and production servers. Relies on `supadmin.bat` to export packages to CTS.	Large	Non-SAP environments.

Package Versioning and Deployment Mode Selection

Unwired Server supports both side-by-side package deployment (or versioned deployment) and replacement deployment. This gives administrators increased flexibility to determine which packages to keep, and when to remove legacy packages.

Versioning varies depending on the role of the deployer, as well as the package type and the deployment method used.

- MBO recommendations for developers:
  - **Deployment mode –** During development, teams typically use replacement deployment, which means the new project completely replaces the existing one. When

the replacement project is deployed, all attributes for the project are replaced in the cache, and project-related tables are purged. As a result, the next time the application synchronizes, all client-side data in the client database is deleted. This reset is recommended for testing, because each deployment instance starts with a clean state.

- **Versioning –** In an MBO project, the version number is used only to find the correct package for deployment. Developers can provide any version number 0 - *n* to deploy successfully.

  By default, developers can keep the version number derived from the version associated with the project. However, if this is an incorrect version, developers can change the default version from the deployment wizard in Sybase Unwired WorkSpace - Mobile Business Object Development, by setting the **Target version** property when performing an update deployment.

- MBO recommendations for administrators:
  - **Deployment mode –** Sybase recommends only using update deployments. For MBO packages, update means the new package follows the backward compatibility guideline defined by Unwired Server. Supported update attribute changes include:
    - Adding MBOs to a new synchronization group
    - Changing the cache policy
    - Adding or removing Result Set Filters
    - Adding or removing Result Checkers
    - Adding or removing a logical role for different role mapping

    When an update is detected by the application, the client database is only updated instead of overwritten. This behavior tightly controls the change process, which is an important factor for production environments.

    If you want to perform a replace deployment, simple delete the MBO first, then redeploy deployment with update mode. All changes are audited by Sybase Control Center.
  - **Versioning –** The administrator cannot change the version number during deployment. Therefore, adjust the version number before deploying the MBO package. To change the version number, open the package, and edit the `deployment_descriptor.xml` file.
- Hybrid Apps recommendations for developers:
  - **Deployment mode –** In Sybase Unwired WorkSpace - Hybrid App Package Development, the deployment mode selection is automatic.
  - **Versioning –** To perform an update deployment, ensure that you set the version number to at least one minor version higher than the version that is currently deployed to Unwired Server. After deployment, both app versions coexist on both the server and device. This allows the user to decide which instance to launch. For a server-initiated Hybrid App, each inbox message (or notification) includes the Hybrid App name and version number. When the user opens the message, the correctly versioned Hybrid App instance is used.

In contrast, for replacement deployments, the version number of the new Hybrid App must match the existing version. The replacement deployment mode then overwrites the deployed Hybrid App. When the newly-deployed Hybrid App is downloaded, the application user launches the Hybrid App with the existing icon; however, the application interface could be completely different.

- Hybrid App recommendations for administrators:
  - **Deployment mode and versioning –** Use New to deploy a new Hybrid App package and its files for the first time. If the uploaded file does not identify an Unwired Server, or an Unwired Server with the same name and version is already deployed to Unwired Server, you see an error message. In the latter case, use an update deployment. Review update mode and versioning behavior outlined for developers. For replace, administrators can replace an existing Hybrid App with a new one, but maintains the same name and version. Administrators typically use the replace deployment mode for minor changes and updates to the Hybrid App.

Deploying Packages with Sybase Control Center

Sybase Control Center is intended for low-volume uploads to Unwired Server. Deploy packages to Unwired Server to test and refine configuration settings in a development environment. After a package has been deployed, configured, tested, and refined in a development Unwired Server environment, it can be transported to a QA or production server environment using export and import. This ensures that the package configuration settings on the source server are moved to the target server. The target server may vary depending on the lifecycle stage your Unwired Platform system supports.

Depending on the package type, not all administrator roles can deploy packages to a pre-production or production instance of Unwired Server:

- For MBO packages, either platform or domain administrator can deploy to Unwired Server
- For Hybrid Apps, only a platform administrator can deploy packages to Hybrid App.

Deploying a Hybrid App Package with the Deploy Wizard

Use the Deploy wizard to make Hybrid App packages available on Unwired Server.

If you are deploying to a target domain, replicate the value in the context variable. The domain deployment target must match the context variable defined. If the developer has used an incorrect context variable (for example, one used for testing environments), you can change the value assigned to one that is appropriate for production deployments.

1. In the left navigation pane of Sybase Control Center, click **Hybrid Apps**.
2. From the **General** tab, click **Deploy**.
3. Click **Browse** to locate the Hybrid App package.
4. Select the file to upload and click **Open**.
5. Select the deployment mode:

- New – deploys an Unwired Server package and its files for the first time.

  If the uploaded file does not contain an Unwired Server, or an Unwired Server with the same name and version is already deployed to Unwired Server, you see an error message.
- Update – installs a new Unwired Server package with the original package name and assigns a new, higher version number than the existing installed Unwired Server package. Sybase recommends that you use this deployment mode for major new changes to the Unwired Server package.

  During the update operation, Unwired Server:
  - Acquires a list of assigned application connections from the original package.
  - Installs and assigns the package a new version number.
  - Prompts the administrator to specify application connection assignments from the acquired list of assigned application connections.
  - Preserves existing notifications.
  - Preserves the previous Unwired Server package version.
- Replace – replaces an existing Unwired Server package with a new package, but maintains the same name and version. Sybase recommends that you use the replace deployment mode for minor changes and updates to the Unwired Server package, or during initial development.

  During the replace operation, Unwired Server:
  - Acquires a list of assigned application connections for each user of the original package.
  - Uninstalls the original package.
  - Installs the new package with the same name and version.
  - Assigns the original application connections list to the new package, thus preserving any application connection assignments associated with the original package.

The package is added to the list of deployed packages, which are sorted by Display Name.

Next

Configure the deployed package if you want it to have a different set of properties in a production environment.

Deploying Native MBO Packages with the Deploy Wizard

Use the Deploy wizard to make MBO packages available on the Unwired Server.

1. In the left navigation pane, click **Domains** > **Packages**.
2. From the **General** tab, click **Deploy**.
3. Click **Browse** to locate the MBO package.
4. Select the file to upload and click **Open**.
5. Select the deployment mode:

- Update – installs a new Unwired Server package with the original package name and assigns a new, higher version number than the existing installed Unwired Server package. Sybase recommends that you use this deployment mode for major new changes to the Unwired Server package.

  During the update operation, Unwired Server
  - Acquires a list of assigned application connections from the original package.
  - Installs and assigns the package new version number.
  - Prompts the administrator to specify application connection assignments from the acquired list of assigned application connections.
  - Preserves existing notifications.
  - Preserves the previous Unwired Server package version.
- Verify – do not deploy package. Only return errors, if any. Used to determine the results of the `UPDATE` deploy mode.

6. If you did not choose a deployment archive as your deployment file, you may browse and select an optional deployment descriptor file.
7. Click **Next** to select the deployment domain security configuration.
8. Click **Next** to configure the mapping state of logical roles of the packge. See *Setting the Mapping State* in *Sybase Control Center for Sybase Unwired Platform*.
9. Review the deployment summary and click **Finish**.

The package is added to the list of deployed packages, which are sorted by Display Name.

Performing Bulk Deployment or Server-to-Server Transports with Utilities or Scripts

Deploy or transport packages with utilities or scripts for high-volume uploads or transfers to Unwired Server. If you have a server with SAP Solution Manager and the CTS plug-in installed, you can use CTS to transport all packages among servers different environments or systems lifecycle stage. In all other environments, use supadmin.bat to perform bulk deployments of multiple packages.

Only a platform or domain administrators can deploy or transport packages among servers.

Deploying Packages with supadmin.bat

Use this utility to perform multiple package administration activities, including deployment, export, and import. You can also use this utility with CTS+ in SAP environments, as part of a transport script.

For complete reference details about this utility, see *Package Administration Utilities* in *System Administration*.

1. To use this utility to deploy a single package from the command line:
   a) From the command prompt change to:
   SUP_HOME`\Servers\UnwiredServer\bin\`

b) Run:

```
supadmin.bat -host host name -port port -u username -pw
password deploy [deploy-options]
```

See *Deploy Command* in *System Administration* for deploy options.

2. To include this utility in a script to deploy multiple files in a batch process, recompile the script after you include all deployment commands.

Deploying Data Orchestration Engine Packages with the SAP DOE-C Utility

Use the **deploy** command in the SAP DOE-C Command Line Utility. Once they are deployed, you can manage these packages with Sybase Control Center.

You must use the **deploy** command in the DOE-C Command Line Utility to deploy a DOE-C package. This is the only DOE-C Command Line Utility command that is not available in the Sybase Control Center.

In an Unwired Platform cluster, deploy the package to the primary Unwired Server node – Unwired Platform automatically replicates the package to the other nodes.

To provide failover and load balancing in an Unwired Platform cluster, specify the URL of a load balancer that is capable of routing to all the Unwired Server nodes in the cluster.

You must specify the URL of a load balancer by setting the **listener.url** property in the `META-INF\sup-db.xml` file, where the ESDMA bundle was unzipped. See in *Mobile Data Models: Using Data Orchestration Engine*.

In the steps below you are prompted for each of the parameters for the deploy command. Alternatively, you may include the parameters when entering the command. See *SAP DOE Connector Command Line Utility* in *System Administration*.

1. Start the SAP DOE Command Line Utility.
   See *Starting the Command Line Utility Console* in *System Administration*.
2. At the **doec-admin** prompt, enter `deploy`, followed by the command parameters.

```
deploy -d|--domain domainName
-a|--applicationID appID
{-u|--technicalUser SAPUserAccount
-pw|--password SAPUserPassword} |
{-ca|-certAlias certificateAlias}
[-sc|--securityConfiguration securityConfigName]
[-dir|--deployFilesDirectory deploymentDirectory]
[-h|--help] [-sl|--silent]
```

Alternatively, enter `deploy`, then enter the parameters as prompted.

Transporting Artifacts Using SAP CTS

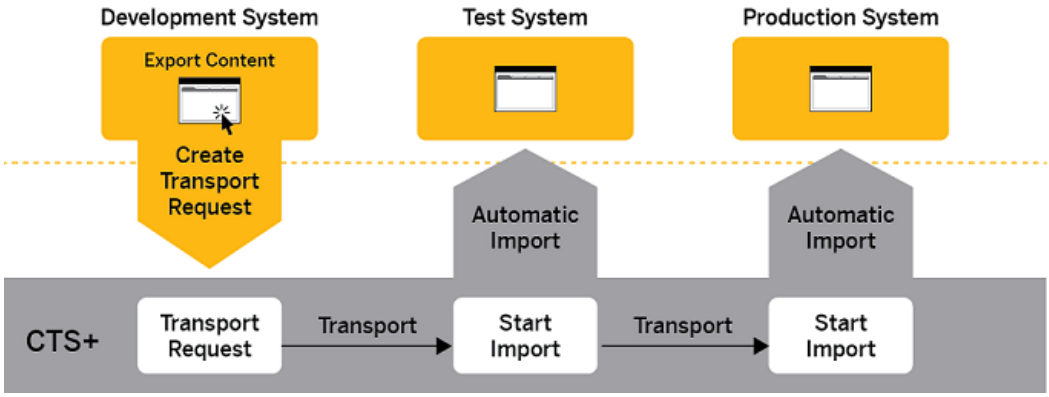
The SAP Change and Transport System (CTS) is a transport management system that enables you to distribute artifacts and automate deployment to different target systems that are connected through transport routes. Administrators can use CTS to distribute Sybase Unwired Platform development artifacts, and automate deployment to different Unwired Servers. For

example, administrators can use CTS to transfer Sybase Unwired Platform artifacts from a development Unwired Server to a test Unwired Server, and then to a production Unwired Server.

Prerequisites

You must have SAP Solution Manager with the CTS plug-in installed on a machine. CTS must be set up and configured for transporting Sybase Unwired Platform artifacts. For details on how to set up, configure, and use CTS, see *CTS Overview* in *System Administration*.

Task



After exporting a package or application from the development system, the administrator creates a transport request in CTS and attaches the export archive. The administrator releases the transport request , and starts the import for the test system. After the import into the test system, CTS forwards the transport request to the queue of the next connected system, the production system. After a successful test, the administrator starts the import for the production system.

Stage 3: Provision

Provision the application to supported devices along with any required artifacts. Provisioning techniques vary depending on the application.

Application provisioning activities include any steps that deliver, install and prepare the application to connect to Unwired Server, and begin the application activation stage. Critical activity includes:

- The distribution and installation of applications and runtime instances as required:
  - (Optional) If you are using Afaria® with Unwired Platform, include an Afaria client for devices supported by the mobile device management solution.
  - An Unwired Platform client runtime for messaging-based synchronization applications for Windows and Windows Mobile. No runtime is required for iOS.
- The seeding of devices with security artifacts for mutual-authentication and encryption, as well as SSO. See *Device Security* in the *Security* guide.
- The seeding of applications with initial application configuration and connection values.

Each of these activities are distinct steps that can be combined as needed to complete the application provisioning stage.

Application Provisioning Tasks By Device

Determine what application provisioning method is available for your device type. If you are provisioning applications to heterogeneous devices, Sybase, suggests that you use Afaria.

Method	Android	BlackBerry	iOS	WindowsMobile
Provision applications and resources with Afaria on page 32.	Yes	Yes	Yes	Yes
Provision applications and resources with native application store. See store documentation for details.	Android Play Store	App World	App Store	
Provision applications and resources with BES on page 14.		Yes		
Provision applications and resources via website download.	Yes	Yes	Yes	Yes
Seeding applications for mutually-authenticated connections on page 44.	Yes	Yes	Yes	Yes

Method	Android	BlackBerry	iOS	WindowsMobile
Pre-configure the application with a configuration file on page 39 . Recommended only for provisioning that is not Afaria-based or for provisioning in preproduction environments.	Yes	Yes	Yes	Yes
Set up push notifications using APNS on page 11.			Yes	
Set up push notifications using GCM on page 19.	Yes			
Setting up BES Environments for Unwired Platform application connections on page 14.		Yes		

Provisioning with Afaria

Enable a group of applications to be initially provisioned with a set of parameters from a provisioning file using Afaria.

Setting Up the Afaria Environment

Setting up the Afaria environment in Unwired Platform primarily involves configuring over-the-air (OTA) deployment. OTA deployment requires specific setup of the Afaria server and uses multiple tools to accomplish this task.

Be aware of the following Unwired Platform requirements:

- Only use IIS on Windows as the OTA Deployment Center host when using Afaria with Unwired Platform.
- Install the OTA Deployment Center behind the DMZ and use a Sybase relay server to relay download requests.
- Do not install the OTA Deployment Center on the same host as either Afaria Administrator or Afaria Server; doing so creates a greater risk of a port conflict, depending on protocols used (by default, the Afaria components share port 80).
- Review the system requirements information Afaria provides.

Setting Up the OTA Deployment Center and the SMS Gateway

The OTA Deployment Center allows device users to access and download Afaria clients and runtimes over-the-air. OTA deployment is recommended for production environments that require you to provision large volumes of applications to heterogeneous devices.

1. OTA Deployment Center requires the Afaria SMS gateway, so ensure this is installed.

See *Afaria Reference Manual | Platform > Server Configuration > Properties > SMS Gateway Installation*.

2. Set up the OTA Deployment Center.

   See *Installing Afaria > Setting up the OTA Deployment Center*. For details about requirements, see *Afaria Release Notes > Component and Feature Requirements*.

Configuring Afaria Server

Configure Afaria Server to use the same gateway as the OTA Deployment Center. You can optionally configure e-mail as well.

1. Open Afaria Administrator and configure Afaria Server to use the OTA Deployment Center and SMS gateway:
   - **Server Configuration > Properties > OTA Deployment Center** to configure settings for this component. If you are using a Relay Server, ensure that you configure OTA Deployment Center to route connections through Relay Server.

     See *Afaria Reference Manual | Platform > Server Configuration > Properties > OTA Deployment*.
   - **Server Configuration > Properties > SMS Gateway** to enable the SMS gateway.

     See *Afaria Reference Manual | Platform > Server Configuration > Properties > SMS Gateway* and *Afaria Reference Manual | Platform > Server Configuration > Properties > Addresses and routing for Afaria messages*.
2. (Optional) Configure an e-mail gateway to send SMS messages over an e-mail infrastructure, which may be used for e-mail-enabled messaging devices.

   See *Afaria Reference Manual | Platform > Server Configuration > Properties > SMTP*.
3. Restart Afaria Server to implement these server configuration changes.

Creating Addresses, Groups, and Profiles

Session Manager allows you to control each user session opened with the SMS or email download notification and specify the actions that are subsequently performed.

The Session Manager channel is used only after the Afaria client files have been correctly installed on the device.

For complete details and references to required related topics, see *Afaria Reference Manual | Platform > Administration > Profile*, *Afaria Reference Manual | Components > Session Manager and Software Manager* chapters.

1. In Afaria Administrator, create address book entries with which to define contact information for devices so that users can be contacted, using the SMS or the email gateway you have enabled.

   Note: If you have a user list in another application and that application allows you to export entries to a `*.CSV` file, you can import this information into Afaria Administrator. See

Afaria Reference Manual | Platform > Home > Client Deployment > Address Book Properties, Distribution List Properties, and Importing Addresses.

2. Create groups, and assign users to them.
3. Create profiles that define what happens during the user session (for example, what instructions are given, which items are sent to a client, and so on).
4. Assign these groups , and therefore its members, to the Session Manager channel.

Create and Deploy Afaria and Unwired Platform Clients

Prepare Afaria and Unwired Platform client deployment. When the user receives a client notification, they download the installer and install the required files for both client types.

1. Use the Afaria Create Client Installation wizard to create a client installer. This program is located only on the Afaria server: `<AfariaServerInstallDir>\Bin\XSClientInstall.exe`.

   A new Afaria client relies on connection configuration settings to connect back to the Afaria Server and run its first session. The Create Client Installation wizard creates seed data and stores it on the client as connection settings.

   CAB files must be signed and placed in the installation sequenced required. Otherwise, the user receives SMS or email messages that link to the OTA Deployment Center in an incorrect order. See *Afaria Reference Manual | Platform > Creating Clients > Creating Afaria Clients*, and refer to the program's context-sensitive help as required.
2. To make downloads available, publish components to the OTA Deployment Center, by using theOTA Publisher, which is installed on the Afaria Server host machine in this location:

   `<AfariaServerInstallDir>\Bin\OTAPublisher.exe`.

   For details about how to use the OTA Publisher, see the program's context-sensitive help.
3. Send a device notification to allow the user to trigger the provisioning process.

   See *Afaria Reference Manual | Platform > Home > Client Deployment > Sending Notifications*. Depending on the gateway you configured, that user address is checked and a message is sent using either SMS or email.
4. Validate performance by checking logs.

   See *Afaria Reference Manual | Platform > Data Views > Working with Logged Actions*.

Launching Afaria from Sybase Control Center

If you purchased Afaria, you can open the Afaria Web interface administration console to manage Afaria resources directly from Sybase Control Center. This makes Afaria a managed resource of Sybase Control Center, and allows you launch and use Afaria remotely.

Prerequisites

Ensure the Afaria Server is running by checking the Windows Services. You cannot open the Afaria Administration console unless this server is running.

Task

1. From the Sybase Control Center menu, click **Resource** > **Register**.
2. Configure the **Resource Type** and **Connection Information** properties.
   For example, to install Sybase Control Center and Afaria Server on the same host, use these properties:

Host	localhost
Port	80

   Otherwise, use the values for the host of the Afaria Server. The Afaria server is added to the **Perspective Resources** list, and takes the display name you configured for it.
3. In the **Perspective Resources** window, right-click the Afaria Server you want to display the administration console for and select **Manage**.

Prerequisites for Provisioning an Application Using Afaria

Verify that these prerequisites for provisioning an application using Afaria are completed by administrators or application developers.

1. Verify that the application developer includes API calls in the application to retrieve the provisioning data and certificate, and design the application user interface to prompt the user for any missing configuration information.

   Note: For the Hybrid Apps, the certificate is retrieved through an API, but the provisioning file is automatically retrieved from Afaria or from a local file.

   See these topics in the appropriate *Developer Guide* for your application type.
   - *Using Afaria to Provision Configuration Data*
   - *Using Certificates from Afaria for Authentication*
2. Developers must share the application ID and required package names used in the configuration file with the administrator so applications can later be activated.
3. Verify that the administrator has configured a Certificate Authority server for use with the Afaria portal package, and has created the public and private keys, and configuration parameters.

Preparing the Provisioning File

Create the provisioning file required for automatic provisioning of an application, and provide it to the Afaria administrator.

1. For each Afaria portal package, create a provisioning file with the settings for connecting the client application to an Unwired Server. See *Creating a Provisioning File*.
2. Deliver the provisioning files to the Afaria Administrator to import them into the Afaria portal package.

Provisioning the Unwired Server Public Key

Use the public key of the Unwired Server to populate its value in the provisioning file.

During start-up, the server verification key is stored in a text file called `ServerVerificationKey.txt` in `SMP_HOME/Servers/MessagingServer`. This file is created only at messaging service startup, so you must start the messaging service to see this file. The file is the Base64-encoded server verification key and contents is used as the value for the `serververificationkey=` key property.

To ensure proper security for application clients on the iOS, Android and Blackberry platforms, seed applications with the Unwired Server public key:

1. Follow the steps of *Seeding Applications for Mutually-Authenticated Connections*. Unwired Platform persists the public key obtained during activation from the trusted intranet connection as the verification key, and validates subsequent requests against this verification key. This key is never exchanged or reset even if the devices are in public networks.
2. To activate the application with a different Unwired Server cluster, the application must clear or reset APIs to clear the verification key, as described in either *Developer Guide: OData SDK* or *Developer Guide: Hybrid Apps*. Alternatively, end users can uninstall and reinstall the application.

Creating a Provisioning File for Afaria

Provision applications through Afaria with a simple text configuration file that can be parsed by the client and from which data is then imported.

1. Create a text file.

   The naming follows a strict naming convention. Use a prefix of `Sybase_Messaging_` followed by the `Application ID`. The file uses `.cfg` extension. Any non-alphanumeric characters in the Application ID are replaced with underscores to avoid any conflicts with operating systems. For example, if the Application ID is “myApplication: 1.0”, the file name is: `Sybase_Messaging_myApplication_1_0.cfg`.

   Note: For Hybrid Apps, settings are applied to the Hybrid Web Container, not to each app.

2. Enter the desired settings information in the `.cfg` file. See *Provisioning Properties Reference*.
3. Save the `.cfg` file.

Provisioning Properties Reference

Enter properties in the provisioning file that seed applications with initial connection registration values to Unwired Server.

The provisioning file is an ASCII text file where each setting must be in a separate line, or separated by a semicolon. All fields in the configuration file are optional—that is, whatever is present in the file is imported.

Key	Description
servername=	The server name for the machine that hosts the Unwired Server or Relay Server if this is used.
serverport=	The server port number for Unwired Server. The default is 5001.
companyid=	The company or farm ID you entered when you registered the device in Sybase Control Center, in this case, 0 (zero).
username=	The name of the user to which the package is assigned. This should be the user name used in the registration.
activationcode=	The activation code for the registered user, for example, `123`.
serververificationkey=	The server verification key, if present, must be a valid Base64-encoded string of the messaging server public key. An invalid Base64 string results in an empty key.

Key	Description
autoreghint=	The preferred method for initial connection registration. The application defaults to the configured method as determined by the value you set. • 0 – No preference. The application implementation decides what is the default method to use • 1 – Default to automatic registration using password authentication • 2 – Default to manual registration • 3 – Default to automatic registration using a provisioned certificate from Afaria. (Afaria Certificate) • 4 – Default to automatic registration using certificate from a local source preferred. (Local Certificate) When an application does not support the requested registration method, an alternative method can be used as determined by the application.
urlsuffix=	The URL suffix is a pattern that is used internally by Unwired Platform when constructing URLs as part of the correct path to connect to Unwired Server. By default, you do not need to set this property; the client auto-detects the correct path when connecting directly to Unwired Server, or when connecting to Unwired Server via the Relay Server using its default settings. **Note:** Only use when the administrator sets something other than the default path and URL suffix for the Relay Server.
protocol=	The protocol used to connect to the Relay Server or the reverse proxy server. (HTTP or HTTPS)

Provisioning Configuration Data and Certificates

Automate the process of provisioning applications by enabling a group of applications to be intially provisioned with a set of parameters from a configuration file, and transferring certificates, using Afaria.

1. The Afaria Administrator performs tasks to configure the Afaria portal package for deploying the provisioning file and X.509 certificate.

Task	Afaria documentation topics
Creates an Afaria portal package.	*Provisioning Configuration Data for an Android Application* *Provisioning a Certificate for a BlackBerry Application* *Provisioning Configuration Data for an iOS Application*
Imports the provisioning file to the Afaria portal package.	
Adds the Afaria portal package to a Group Profile.	
Configures a Certificate Authority server for the Afaria portal package.	*Provisioning a Certificate for an Android Application* *Provisioning a Certificate for a BlackBerry Application* *Provisioning a Certificate for an iOS Application*

2. The Afaria Administrator distributes the application through Afaria server, and instructs the user how to connect to the Afaria environment and enroll in management. User installs the application through Afaria client.
   When the application starts, the application prompts the device user to enter the required configuration information for the CA server, such as the Common Name and Challenge Code.

Provisioning Applications with Configuration Files

In environments where Afaria is not used for mobile device management, you can use a configuration file to provision the device application on BlackBerry, Android devices, or Windows Mobile or Win32 devices. You cannot manually provision iOS clients in this manner.

When the application first starts, the client searches for the configuration file on a flash card or at a fixed platform-specific location.

Provisioning the Unwired Server Public Key

(Optional) Retrieve the public key of the Unwired Server to populate its value in the provisioning file.

During startup, the server verification key is stored in a text file called *SUP_HOME*`\Servers\MessagingServer\ServerVerificationKey.txt`. It is created only at messaging service startup, so you must start the messaging service once to generate this file. The file is Base64 encoded, and the contents must be used as the value for the `serververificationkey=` key.

Creating a Provisioning File

Provision applications with a simple text configuration file that can be parsed by the client and from which data is then imported.

The provisioning file is an ASCII text file where each setting must be in a separate line, or separated by a semicolon. All fields in the configuration file are optional—that is, whatever is present in the file is imported.

1. Create a text file.

   The naming follows a strict naming convention. Use a prefix of `Sybase_Messaging_` followed by the `Application ID`. The file uses `.cfg` extension. Any non-alphanumeric characters in the Application ID are replaced with underscores to avoid any conflicts with operating systems. For example, if the Application ID is "myApplication: 1.0", the file name is: `Sybase_Messaging_myApplication_1_0.cfg`.

   Note: For Hybrid Apps, settings are applied to the Hybrid Web Container, not to each app.

2. Enter the desired settings information in the `.cfg` file. See *Provisioning Properties Reference*.
3. Save the file.

Provisioning Properties Reference

Enter properties in the provisioning file that seed applications with initial connection registration values to Unwired Server.

The provisioning file is an ASCII text file where each setting must be in a separate line, or separated by a semicolon. All fields in the configuration file are optional—that is, whatever is present in the file is imported.

Key	Description
servername=	The server name for the machine that hosts the Unwired Server or Relay Server if this is used.
serverport=	The server port number for Unwired Server. The default is 5001.
companyid=	The company or farm ID you entered when you registered the device in Sybase Control Center, in this case, 0 (zero).
username=	The name of the user to which the package is assigned. This should be the user name used in the registration.

Key	Description
activationcode=	The activation code for the registered user, for example, `123`.
serververificationkey=	The server verification key, if present, must be a valid Base64-encoded string of the messaging server public key. An invalid Base64 string results in an empty key.
autoreghint=	The preferred method for initial connection registration. The application defaults to the configured method as determined by the value you set. • 0 – No preference. The application implementation decides what is the default method to use • 1 – Default to automatic registration using password authentication • 2 – Default to manual registration • 3 – Default to automatic registration using a provisioned certificate from Afaria. (Afaria Certificate) • 4 – Default to automatic registration using certificate from a local source preferred. (Local Certificate) When an application does not support the requested registration method, an alternative method can be used as determined by the application.
urlsuffix=	The URL suffix is a pattern that is used internally by Unwired Platform when constructing URLs as part of the correct path to connect to Unwired Server. By default, you do not need to set this property; the client auto-detects the correct path when connecting directly to Unwired Server, or when connecting to Unwired Server via the Relay Server using its default settings. **Note:** Only use when the administrator sets something other than the default path and URL suffix for the Relay Server.
protocol=	The protocol used to connect to the Relay Server or the reverse proxy server. (HTTP or HTTPS)

Distributing and Loading the Application Provisioning File

Make the provisioning file available to the device so that it can be automatically loaded when the application starts.

When the application is started, a search is performed for the provisioning file only if the device is not yet configured; that is, missing one or more of the required connection properties (server name, port, user name, activation code, and so on). If the application is unconfigured, the settings from the provisioning file are automatically applied.

Deliver the application provisioning file to the device and make it available to load when the application starts.

Methods of delivery vary:

- **Android –** use e-mail or messaging to upload the provisioning file to the root of the flash card (`/sdcard/`), or the client file root (`/data/data/com.sybase.workflow/files`).

  Note: Some Android devices, for example, Samsung, have built-in storage with the folder name "sdcard," so even if there is not an SD card inserted in the device, you can still see an `sdcard` directory. The provisioning file does not work in the `sdcard` directory unless there is an SD card inserted in the device.
- **BlackBerry –** use BES push, e-mail, or messaging to upload the provisioning file to the root of the flash card (`fileconn.dir.memorycard`), or the client file root (`file:///store/home/user/`).
- **iOS –** Afaria client deployment.
- **Windows Mobile –** use e-mail or messaging to upload the provisioning file to the client root of any flash cards, or the root of the file system.
- **Win32 –** use e-mail or messaging to upload the provisioning file to the Application Data subdirectory which resides in the Special Folder path defined by CSIDL_APPDATA (`%APPDATA%\Sybase\MOMessaging`).
- **Hybrid Apps –** Delivery is not applicable. Searches and loads files by as previously described for each operating system.

Provisioning with OS-Specific App Stores

Different devices offer OS-specific app stores you may be able to provision devices with. Each store may have different requirements and limitations you need to accommodate before being able to post an application for self-service download.

See the documentation for these stores as required:

- For BlackBerry, see the App World documentation.
- For Android, see the Android Play Store documentation.
- For iPhone, see the App Store documentation.

Provisioning with Native Options for BlackBerry Devices

To provision the application to BlackBerry devices, you can automatically push the application to the device or send a link to device users so they can install it when desired. For small deployments or evaluation purposes, device users can install the application using BlackBerry Desktop Manager.

Once installed on the device, the application appears in Downloads. However, device users can move it to a different location. If device users reinstall the application from a link or URL, or using Desktop Manager, the BlackBerry device remembers the installation location.

Provisioning Method	Used for	Description
BlackBerry Enterprise Server (BES) over-the-air (OTA)	Enterprise installations	When the BlackBerry device activates, it automatically uses BES to download the application. For BES MDS 5.*x* step-by-step instructions, see *Sending Software and BlackBerry Java Applications to BlackBerry Devices* in *BES Administration Guide* located at: *http://docs.blackberry.com/en/*. The `HybridWebContainer.zip` file needed for deployment is located in *`SUP_HOME`*`\MobileSDK<version>\HybridApp\Containers\BB\BES Deployment`. For BES MDS 4.*x* step-by-step instructions, see *Managing the Delivery of BlackBerry Java Applications, BlackBerry Device Software, and Device Setting to BlackBerry Devices* in *BES Administration Guide* located at: *http://docs.blackberry.com/en/*. See *http://www.blackberry.com/btsc/search.do?cmd=displayKC&docType=kc&externalId=KB03748*.

Provisioning Method	Used for	Description
OTA: URL/link to installation files	Enterprise installations	The administrator stages the OTA files in a Web-accessible location and notifies BlackBerry device users via an e-mail message with a link to the `CRMUI.jad` file. The `.JAD` and associated files for this type of deployment are located in *`SUP_HOME`*`\MobileSDK<version>\HybridApp\Containers\BB\OTA`.
Desktop Manager	Personal installation	Installs the application when the BlackBerry device is synced via a computer.
BlackBerry App World	Evaluation/Demo	Downloads a solution-specific application in demo mode. Demo mode runs standalone, and the device does not connect to Unwired Platform. *http://appworld.blackberry.com/webstore/*

Provisioning the Public RSA key from the Messaging Server for MBS Encryption

If you are not using Afaria, you can install the client application, then connect to the corporate LAN using Wi-Fi or other method of your choosing in order to provision devices with required files. This allows you to seed public RSA keys to the device so that over-the-air connections to Unwired Server can be mutually-authenticated and you can minimize the possibility of a rogue server intercepting your initial synchronization and providing its own RSA public key.

Follow these steps to ensure that the public RSA key required for future secure communication is correctly and reliably installed.

1. Provision the application to the device.
2. Connect to the corporate LAN on which the Unwired Server cluster is installed.
3. Use a device connection that connects directly to Unwired Server. Alternatively, you can also connect using the Relay Server settings, but only if Relay Server is accessible from the corporate LAN; typically it is deployed on the DMZ
   Unwired Server seeds the client with the public key. The client uses this public key for all subsequent connections.
4. Provide the user with instructions to reconfigure the connection properties on the device to use Relay Server from the Internet for subsequent connections.

Stage 4: Activate

The application activation stage defines all activities that relate to the start-up of and formal entry of an application and a known or anonymous user into the Unwired Platform runtime. It ties all entities together by associating a user with the application and its resources (connections, customizations, and packages).

Note: Agentry applications have different steps in the Activate stage. For more information, see *Creating Agentry Application Definitions* in *Sybase Control Center for Sybase Unwired Platform*.

The activation process is dependent on a one-time handshake/setup between the application and Unwired Server. This handshake is dependent upon the device being preseeding with required configuration and security artifacts. These artifacts and configuration values may come from one or more sources, such as the user, a provisioning file installed with the application, or delivered OTA with Afaria. See the *Stage 3: Provision* chapter for details.

After an application becomes activated:

- **Unwired Platform client applications –** can access all Unwired Platform features, including complete data synchronization, push notifications, and so on.
- **HTTP open service client applications –** can only access a subset of Unwired Server services.

The Activation Process

Various features support the application activation process. Application types (for example, native applications or Hybrid Apps), and design decisions (for example, connection registration type, customizations) each might have a significant impact on the administration activities required to support your environment.

The pairing of users to applications, and applications to connections, packages, and customizations, works only if all entities are recognized. Key application activation features that support this pairing include: application definitions connection registration (manual, automatic, or anonymous) properties, each of which contain fundamental details that allow the application to become activated.

How Applications Are Recognized

Administrators or developers must uniquely identify applications with an application ID. No two applications may share a common application ID.

Administrators can define this ID in Sybase Control Center when an application is defined. This definition of the general application properties allows the connecting application to be recognized by Unwired Server.

The application ID allows:

- General application settings to be defined for the application
- Administration of the application in Sybase Control Center
- Application packages to be located
- Users to be paired with the application (manually, automatically, or anonymously)
- (Hybrid Apps) Multiple Hybrid Apps to be grouped.
- (Hybrid Apps) Subscription per package user for each app to be assigned.

How Connections Are Registered

Connection registration occurs after the user, device, and application are recognized and assigned to a connection. This multi-way grouping creates a unique connection in Sybase Control Center, so the registered connection activity can be monitored.

The process by which a user becomes recognized depends on whether:

- The real identity need to be established with by Unwired Server or by the EIS data source. Sybase recommends using an identity to recognize the user.
- The degree of manual intervention required to complete registration.

Once the connection is registered, the platform administrator can view the application user that is associated with the activated application connection; however, the user name displayed is not necessarily same as an authenticated user identity. Consider:

- Only automatic registrations use the extracted user identity.
- For manual activation code-based registration, the administrator can freely name the user. The relationship between user name and application connection is established when you click the **Package Users** button for an in-bound connection.
- No user names appear for anonymous role-based registration.

Note: SAP applications that have connections registered with Unwired Server, can also have licenses counted by SAP License Audit service. For a list of SAP applications for which licenses are counted, see *SAP Applications Tracked with SAP License Audit* in *System Administration*.

Automatic Connection Registration with Identities

Automatic connection registration is the process by which a user and application ID is paired to an connection by using the authentication identity. It is considered automatic, because the administrator does not need to manually create this pairing by selecting the in-bound connection and registering it in Sybase Control Center.

For automatic connection registration, administrators must create a connection template and use it with the application definition. The combination application ID, domain, and security property values must be unique.

Note: For messaging clients, automatic registration will not work if the user name contain i18n characters.

A connection that is registered allows the application to become activated (that is, allowing it to consume data and services of Unwired Platform). The activation process for automatic connection registration follows this sequence:

- **Login –** the user sends credentials via the application the security provider.
- **Authentication –** the security provider delivers the authentication result to Unwired Server.
- **Connection registration –** the user identity is extracted (which may include a security context as in user1@SecurityConfiguration1) and paired with a device ID and one or more applications IDs.
- **Connection activation –** the registration information identifies required packages, customizations, and connections on Unwired Server. If the **Automatic Registration Enabled** property for the connection is set to `True`, then the template automatically creates a default connection for the application on Unwired Server.
- **Application activation –** the device synchronizes settings from the server, and returns device information back to the server. For example, the registered application connection is updated with a phone number, IMSI, device type, and perhaps native push configuration information that is used to deliver notifications via device-specific infrastructures for Apple, BlackBerry, or Android.

Manual Connection Registration with Activation Codes

Activation codes are an alternate way to complete the connection registration process. It is considered a manual registration method because users must enter the supplied activation code before administrators can manually register the inbound connection request.

For online applications, manual registration also includes a security token (x-sup-sectoken) in the header value, which represents the same activation code in a hashed form.

A connection that is registered allows the application to become activated (that is, allowing it to consume data and services of Unwired Platform). The activation process for manual connection registration follows this sequence:

- **Delivery –** the administrator delivers the activation code; for example, phone SMS, e-mail, or Afaria.
- **Login –** the user sends credentials via the application the security provider.
- **Code entry –** the activation code initiates the activation process. Once delivered, the code helps find packages, customizations, and connections on Unwired Server. The activation code is a one-time use, limited-duration (by default, 72 hours) code. Once the expiry time is met, the ability to activate the application expires, not the activation code itself. If users miss the activation period, the must submit a request to the administrator to use a new code and re-try again.
- **Connection registration –** the administrator registers the connection by selecting the in-bound connection and manually pairing it to a template. The selected template supplies the default connection values. The administrator must also manually enter an identity for the application user of the registered connection in Sybase Control Center.

- **Application activation –** the device synchronizes settings from the server, and returns device information back to the server. For example, the registered application connection is updated with a phone number, IMSI, device type, and perhaps native push configuration information that is used to deliver notifications via device-specific infrastructures for Apple, BlackBerry, or Android.

Note: You can authenticate users after application activation. This authentication can subsequently occur, for example, when an application tries to synchronize (as in the case for native applications) or makes some other online request (as in the case for Hybrid Apps). Any security configurations you have used to authenticate a package is then used to authenticate the request.

Anonymous Connection Registration

(Recommended only for online applications) In large business-to-customer deployments, application design may support anonymous user connections to Unwired Server.

Anonymous access can be configured for application connections to Unwired Server or for EIS connection pools on the enterprise back-end. They are authenticated using an anonymous security configuration. All anonymous users are identified with an anonymous role when Unwired Server receives the initial connection based on the configuration set for the application.

However, because the connection is not registered with a unique user identity, the connection does not appear in the activated application connection list in Sybase Control Center. The connection is automatically activated once the in-bound connection request is processed by Unwired Server.

Note: If an offline application uses an MBO operation that relies on a SAP BAPI function, the anonymous user may not be sufficient to perform the operation in the EIS. Sybase recommends that you include a fixed user configured inside the SAP endpoint to access backend system. When the developer designs the MBO load operation, the developer must use a non-client credential. This configuration ensures that the MBO uses the credential configured in the endpoint.

When Reregistration is Required

Because the assignment of connections and resources occurs after applications, identities, and devices are paired, then reregistration is required when this association breaks or requires a different pairing.

Reregistration is required, for example, if an application user loses a device, or when devices are shared among multiple users.

Not all applications support reregisteration. To determine if the client application supports reregistration, confirm that the value in the **Application Supports Client Callable Components** connection property on the **Capabilities** tab is set to True.

See *Registering or Reregistering Application Connections* in *Sybase Control Center for Sybase Unwired Platform*.

How Applications are Activated

Applications' resources are assigned only once the initial in-bound connection is registered, and the user becomes a formal application user, and recognized by Sybase Control Center.

Connection registration:

- Pairs the initial inbound connection with a user and an application instance by registering this information with Unwired Server
- Triggers formal entry into the Unwired Platform runtime with the assignment of connection and customization resource bundles. Once assigned, the application is considered activated as it is directly linked to Unwired Server and interacts with its services, thereby allowing it to consume required data.

Application Connections

(Not applicable to Agentry applications) Application connections define the manner in which the application connects to Unwired Server and interacts with the runtime services the application requires. Connections are defined by the properties that are configured for it, frequently saved as a reusable template.

The connection registration process is either manual or automatic. Without successful registration, a connection is not activated, and an application cannot consume the data, the customizations, or the packages it requires.

Sybase recommends that you use templates to store and reuse connections. At minimum, you should create one template per security configuration.

Application Customization Resource Bundles

(Does not apply to Hybrid App or Agentry SDK clients) For supported application types, customization resource bundles enable you to associate deployed client applications with different versions of customization resources.

A customization resource bundle is a JAR file that includes a manifest file of name and version properties. The customization resource bundle does not contain any information that binds or helps bind to applications; it can be uploaded or exported during the definition of an application with Sybase Control Center. A deployed customization resource bundle is read-only.

Implementing a customization resource bundle requires the coordination of various roles:

1. (Application developer) Invokes the SDK API that downloads the customization resource bundle. Use the `onCustomizationBundleDownloadComplete` (in the `Application Callback API`) and `BeginDownloadCustomizationBundle` (in the SUPApplication class) methods to pair the application with the device, and reach the client application. See *Developer Guide: OData SDK* or any of the Object API *Developer Guides*.
   For example, for an application called `Sybase.Mobile.Application`, you might implement the customization resource bundle invocation as follows:

```
/// <summary>
/// start downloading default resource bundle associated with the
application.The resource bundle would be saved into writer stream
provided by user.
///an application only bundle an resource
///</summary>
///<param name="writer">a writer stream provided by user
///</param>
public void BeginDownloadCustomizationBundle(System.IO.Stream
writer) { }

/// <summary>
/// start downloading resource bundle named customizationBundleID.
The resource bundle would be saved into writer stream provided by
user.
///</summary>
///<param name="customizationBundleID">the resource bundle name
///</param>
///<param name="writer">a writer stream provided by user
///</param>
public void BeginDownloadCustomizationBundle(string
customizationBundleID, System.IO.Stream writer) { }
Sybase.Mobile.IApplicationCallback

/// <summary>
/// Invoked when download resource bundle complete.
/// </summary>
/// <param name="customizationBundleID">! the resource bundle
name. if null, application default resource bundle is downloaded
/// </param>
void OnCustomizationBundleDownloadComplete(string
customizationBundleID, int errorCode, string errorMessage);
```

2. (Developer) Generates the `JAR` with the `MANIFEST.MF`, which includes these required properties:
   - `Customization-Resource-Bundle-Name`
   - `Customization-Resource-Bundle-Version`
3. (Administrator) Uses Sybase Control Center to upload the customization resource bundle to Unwired Server then assign it to an application connection.
4. Once the application activation process completes, the application is directed to the appropriate version of the resource bundle.

Application Customization Resource Bundle Recommendations

There are a variety of recommendations for working with customization resource bundles.

- You cannot use customization resource bundles for Hybrid App and Agentry SDK clients.
- The expected format of the customization resource bundle is a JAR archive that contains `MANIFEST.MF`.
  - The manifest file must include these properties:
    - `Customization-Resource-Bundle-Name`

    - `Customization-Resource-Bundle-Version`
  - Property values cannot include a colon (":").
  - File size should not exceed 5MB. File size is not enforced, but the larger the file, the slower the performance, and can be subject to device platform hardware capabilities.

  See *Managing Application Customization Resource Bundles* in *Developer Guide: Unwired Server Runtime* for information about the administration API that allows programmatic access to this functionality.
- You can assign the same customization resource bundle to different application connections, and it is treated independently for each application that is paired with the connection template that identifies the bundle and version. The primary key is: application ID, customization resource bundle name, and version.
- Each customization resource bundle:
  - Belongs to one and only one application. If you delete an application, all associated customization resource bundles are deleted as well. This implies that the actual binary is stored twice when assigned to two application IDs.
  - Is applicable only to:
    - The application to which it belongs.
    - The application connections that have the same application ID.
    - The application connection templates that have the same application ID.
  - Takes effect only when it is assigned to one or more application connections.
  - Is by default, not assigned to either application connections or application connection templates.
  - Must be assigned explicitly by configuring an application connection or application connection template to use a customization resource bundle.

    Note: The application connection assignment configuration overrides that of the application connection template.
  - Can be exported to the same JAR file being uploaded to Unwired Server, meaning the format does not change.
- Can upload more than one customization resource bundles, as long as the name and version combination is unique.
- Can assign only one primary customization resource bundle in an application definition. However, any uploaded customization resource bundle is accessible to any application connection.
- You can delete a customization resource bundle only if it is not assigned to any application connection and application connection template with the same application.

Best Practices for Avoiding Common Registration and Activation Errors

Avoid common problems encountered when registering connections, and activating applications once the connection is registered.

- **Application activation –**
  - The value of the Server address, Relay Server address, Port, and Company ID, sent from the application (device) must all be correct – if not, when registering an application to Unwired Server, an error (in this case for an incorrect Relay Server address) similar to:

    ```
    Error: 558 Message: 'Could not connect to server.
    Verify Relay Server URL Template'.
    ```

    is returned.
  - The value of the username or password/activation code sent from the device must be correct – if not, the error:

    ```
    Error: 580 Message: 'TM Error:InvalidAuthenticationParameters'.
    ```

    is returned when attempting to activate/register the application.
  - For BlackBerry, Android, Windows Mobile, and iOS native object API applications, if an application is deleted from Sybase Control Center, when the client application calls `startConnection()`, the following callback is triggered inside the `ApplicationCallback` handler:

    ```
    void onConnectionStatusChanged(int connectionStatus, int
    errorCode, String errorMessage);
    errorCode = 580
    errorMessage = "Error: 580 Message: 'TM
    Error:InvalidAuthenticationParameters'"
    ```

    To continue using the application, unregister and re-register the application. You lose the previous subscription on the server side. Delete the client database and perform another initial synchronization. See the topic *startConnection* in the Developer Guide for your platform for details.
  - Use the `clearServerVerificationKey()` method in your application to avoid registration problems when moving from one server to another – when you have registered a device on one Unwired Server, then try to connect the same device to a different Unwired Server, you may encounter the error:

    ```
     584: Wrong public key
    ```

    if you do not use `clearServerVerificationKey()` in ODP applications.
  - During initialization, the client may try several URLs in an attempt to discover the correct path to the Unwired Platform server. During these attempts, 404 errors `Page Not Found` may occur and are reported to the OnHttpCommunicationError callback. This is expected behavior and you can safely ignore these initial 404 errors.

If the 404 errors continue to occur after successful application activation or if device registration never completes, verify the UrlSuffix setting of the application's ConnectionProperties object

- **User or device registration –**
  - If a physical device is already registered and is to be reregistered in Unwired Server, you must first delete the existing device through the available API, then reregister – if you attempt to register an already registered device without first deleting the device, the error:

    ```
    14853: Wrong user name for this device.
    ```

    is returned.

    Note: The error:

    ```
    14899: Unknown error.
    ```

    can be returned for a variety of reasons, including the same registration problem that generates error code 14853.
  - Use correct credentials when registering single-sign on (SSO) based registration – if you use incorrect username or password while carrying out SSO based registration, the error:

    ```
    14814 Message: MMS Authentication Failed .
    ```

    is returned.
  - The security configuration on the device and Unwired Server must match – during registration, the error:

    ```
    14850 Message: Auto registration template not found .
    ```

    is returned when the security configuration passed from the application installed on the device does not match that for the application in Unwired Server.

Enabling Application Activation with Sybase Control Center

Application activation is the end result of the series of events by which a user can start consuming data or using platform services. Application activation is enabled by the administrator using Sybase Control Center, then triggered by the application user upon a first connection attempt.

Prerequisites

Deploy packages and seed devices before the application initiates a connection. See *Stage 2: Deploy* and *Stage 3: Provision* for details.

Task

The process by by which activation occurs varies depending on the type of connection registration used: automatic, manual, or anonymous. However, administration activities comprise a key set of tasks.

Creating and Assigning a Security Configuration

Prepare a security configuration that one or more administrators can select as part of the application definition or connection template. The security configuration is used in different ways, depending on whether you are performing manual or automatic application user registration.

Create as many security configurations as needed, depending on the number of systems to be accessed. Sybase recommends that you have as many connection templates as there are security configurations (a 1:1 ratio).

- **For manual registration –** For manual registration, the security configuration is not used during the activation process. However, it is use after the application is activated to authenticate operations the application makes over the assigned connection.
- **For automatic registration –** the security configuration you create authenticates the user identity, and passes an identity under which the user is registered and the session ID assigned. To use a security configuration for this method, ensure automatic registration is enabled in the connection template.
- **For role-based access –** if applications have different security requirements, use logical roles to implement a more fine-grained control to applications. You can use role-based access control for both manual and automatic registration.

1. In Sybase Control Center, create a security configuration at the cluster level. Add at least one security provider to it, setting the Control Flag attribute to an appropriate value for the number of providers you add.

   The type of provider you choose depends on:
   - The security repository you use in your production environment.
   - Whether you need to support SSO and what type of credential will be passed.
   - The EIS used. For example, SAP connections, Sybase recommends that you use either a SAPSSOTokenLoginModule, a CertificateAuthenticationLoginModule, or a HttpAuthenticationLoginModule.

   For details, see *Creating a Security Configuration* in *Sybase Control Center for Sybase Unwired Platform*.
2. Create and map the logical roles needed to control access to the application connections that are associated with this security configuration. If no mappings are created, all authenticated users can access the connection.

   For Hybrid Apps, the logical role you define for the application should be the same logical role used for package deployment. The role chosen during package deployment allows the users who are authorized by the logical roles to install the package.

3. If you require any specific authentication cache behavior, click the **Settings** tab, and configure the properties as required.
4. Save the changes.
   The security configuration now appears in the list of available security configurations when you configure properties for an application connection. Users can now access required connections only after being properly authenticated and appropriately authorized via the roles defined and mapped. Authentication allows the administrator to review:
   - The users who are accessing an application and determine which security configuration was used to grant access to the application.
   - The application connections that are actively used by users and what role was used to confer active status.
   - The applications to which authenticated users have access. .

Uploading Application Customization Resource Bundles

(Does not apply to Hybrid App or Agentry SDK clients) Before you can assign application customization resource bundles, you must upload them to Unwired Server with Sybase Control Center.

Only platform administrators can upload bundles to Unwired Server.

1. (Optional) Click **Refresh** to update the list of deployed customization resource bundles for this application.
2. (Optional) Add another customization resource bundle to the list.
   a) Click **Add**.
   b) In the file dialog, navigate to and select the customization resource bundle JAR file, and click **OK**. The name and version of the newly deployed customization resource bundle is added to the list.
   c) (Optional) In the Confirm dialog, select one or more check boxes to assign the newly uploaded bundle to application connections or application connection templates with the same application ID. If no check boxes are selected, there is no automatic assignment.

      Note: You can make these assignments at a later time.

   d) Click **OK**.
3. (Optional) Add another customization resource bundle to the application.
4. Click **OK**.

Creating Application Connection Templates

If you have a set of connection properties you want to reuse with multiple applications, create a template before you create an application definition. You can later override default template values in the template during the application definition phase as needed.

1. In the left navigation pane of Sybase Control Center, click the **Applications** node.

2. In the right administration pane, click the **Application Connection Templates** tab, then **New**.
3. Enter the name and description for the application connection template.
4. Select a base template.
5. (Optional) Override properties of the template, if required.

   Recommendations vary depending on the device type, the application type, and the type of application connection registration required (manual, automatic, or anonymous). For details about all other properties, see *Application Settings* in *Sybase Control Center for Sybase Unwired Platform* online help.
   - **BlackBerry devices –** to enable native push notifications for these devices, configure the BlackBerry Push Notifications tab.
   - **iOS devices –** to enable native push notifications for these devices, configure the Apple Push Notifications tab.
   - **Android devices –** to enable native push notifications for these devices, configure the Android Push Notifications tab.
   - **Manual connection registration –** has multiple requirements:
     - In the Application Settings tab, verify that automatic registration is disabled: set **Automatic Registration Enabled** to false, and do not select a security configuration. A logical role is not required, because no application-level authentication or authorization is performed.
     - In the User Registration tab, ensure that values are configured to an appropriate value to support activation code-based registration. The length of the code you set here must match that used in the Connection Properties tab.
     - In the Connection Properties tab, set a unique activation code for the application and communicate that code to all users of that application.
   - **Automatic connection registration –** in the Application Settings tab, check that automatic registration is enabled: set **Automatic Registration Enabled** to true. Select a security configuration and choose the logical role to which application users must belong to before access is granted and the connection activated.
   - **Anonymous connection registration –** has multiple requirements:
     - In the Application Settings tab, select a the Anonymous security configuration.
     - For ODP, in the Proxy tab, select Allow Anonymous Access.
     - (Optional) If parts of the application require subsequent authentication and authorization, create a separate security configuration and use the logical role for a different template.
   - **Hybrid Apps –** ensure that you assign packages to the connection after it has been configured.
6. Click **OK**.

You can assign the connection template to an application when you are configuring values for an application definition.

Application ID and Template Guidelines

Choose an appropriate application ID while registering application connection for use by native MBO, Hybrid App, or Online Data Proxy clients. Using an incorrect application ID results in failure when the client tries to activate itself.

Application Type	Guidelines
Hybrid App	• 2.0.1 or earlier – leave the application ID empty. • 2.1 or later – use preexisting HWC template, or, if you are using your own template, make sure that HWC is set as the application ID in the template. • iOS sample container 2.1 or later – use the template you have created. The application ID used by the iOS sample container should match the application ID specified in registration.
Native MBO application	• Previous to 2.1.2 – leave the application ID empty. This applies to native messaging-based application clients. • 2.1.2 or later – (recommended) use the application connection template that is automatically created for the application. Otherwise, ensure you register the application connection with the correct template by verifying that application ID matches, and that the correct security configuration and domain are selected. Also, if using replication, set other template properties (such as synchronization-related properties in Connection category) as required. For Android native MBO applications, this recommendation applies starting with version 2.1.1.
Online Data Proxy	Register the application connection using the template created for the application. Existing templates can be found in the **Applications > Application Connection Template** tab.

Subscribing MBO Applications to Push Synchronization Notifications

(Not applicable to Online Data Proxy) If your device application requires push synchronization, you must configure the application accordingly. Push notifications for synchronization requires coordination between developers and platform administrators.

1. The developer creates the application using MBO packages. The application must be modeled for push synchronization using available callbacks. See *Mobile Data Models: Using Mobile Business Objects*.
2. The administrator creates one or more subscription templates for each MBO package used in Sybase Control Center. See *Creating Replication Subscription Templates* in *Sybase Control Center for Sybase Unwired Platform* online help.

Subscriptions are processed differently depending on the type of application:

- **For replication applications –** When the application user synchronizes the package, a subscription is created on Unwired Server. Subscription properties (derived from the template) enable Unwired Server to send notifications when MBO data changes are detected. The change detection frequency is based on the synchronization group property to which the MBO belongs. Developers can push EIS data updates to the cache using DCNs. Administrators can configure cache groups to automatically refresh the cache per the schedule defined.

  When Unwired Server identifies data changes, it generates data notifications for the client, and sends them according to the client's subscription properties. Delivery occurs when the application is online and connected to Unwired Server. The client application developer is expected to handle the notification by initiating a synchronization request to pull the changes down (with or without user prompt).
- **For messaging applications –** Messaging subscriptions are created upon receiving subscription request from the client application and data synchronization messages are sent to the device.

Creating an Application Definition

An application definition is required for all applications.

Determine whether or not you need to set an application ID or whether one is already assigned to the application. For example, for Hybrid Apps, developers can preconfigure this ID.

1. In Sybase Control Center, click the **Applications** node and select the **Applications** tab in the right administration pane.
2. Click **New**.
3. Set the general properties for the application:
   - An application ID, if one is not already set. See *Setting General Application Properties* in *Sybase Control Center for Sybase Unwired Platform* for application ID guidelines.
   - A display name and an optional description.

- (Recommended) Select a template and override template properties as required. Otherwise, you can configure properties for single application use only. For Hybrid Apps you can use the HWC template that exists for this application type.
- The domain to which the application is assigned.
- (Optional) Assign one or more packages to the application.

4. Click **Finish** to complete the definition.

Assigning and Unassigning a Hybrid App to an Application Connection

Assign a Hybrid App package to an application connection to make it available to a device user. Unassign the Hybrid App package when it is no longer required.

You can also assign Hybrid App packages indirectly through the application connection template. The set of packages assigned to an application connection will be a combination of packages assigned indirectly through the application connection template and directly through the application connection.

1. In the left navigation pane of Sybase Control Center, click **Hybrid Apps** and select the Hybrid App to assign.
2. In the right administration pane, click the **Application Connections** tab.
3. Click **Assign**.
4. List the activation users to assign the Hybrid App package to.

   Search for users by selecting the user property you want to search on, then selecting the string to match against. Click **Go** to display the users.
5. Select the user or users from the list that you want to assign the Hybrid App package to.
6. Click **OK**.
7. To set the Hybrid App package as the default application for an application connection, select the connection and click **default**.

   Set a Hybrid App package as the default to run that application on the device as a single-purpose application. Single-purpose applications launch automatically when the user opens the Hybrid Web Container. This will be the only Hybrid App available on the device. You can select only one default per application connection.
8. To unassign a Hybrid App package, select the application connection and click **Unassign**.

   Note: If you unassign the Hybrid App package that is set as the default, you may want to select a new default package.
9. Click **OK**.

Setting and Unsetting a Default Hybrid App

Set a Hybrid App package as the default application for an application connection to run it on the device as a single-purpose application. A single-purpose application launches

automatically when the user opens the Hybrid Web Container and is the only Hybrid App available on the device.

1. In the left navigation pane of Sybase Control Center, click **Applications**.
2. In the right administration pane, click the **Application Connections** tab.
3. Select the application connection to set the default for.
4. Click **Hybrid Apps**.
5. Select the Hybrid App package to set as the default and click **Set default**.

   You can select only one default per application connection.

   To remove a Hybrid App package as the default, select the package and click **Unset default**.
6. Click **OK**.

Single-Purpose Hybrid Apps

Single-purpose Hybrid Apps launch automatically when the device user opens the Hybrid Web Container.

To make an application single-purpose, you must set it as the default for the device. If the device has only one assigned Hybrid App, but you do not set it as the default, the user must navigate through the Hybrid Web Container and select the Hybrid App.

Once you have set a default, you can still assign other Hybrid Apps to the device, but only the default application is active.

You can change the default application at any time if you want the user to see a different application when the Hybrid Web Container is launched. For example, for an application that shows coupons or deals, you can update the deals each week and assign a new application with the updated deals as the default.

Registering Initial Application Connections

(Required only for manual connection registrations) Use Sybase Control Center to manually register an in-bound application connection.

1. In the left navigation pane, click the **Applications** node.
2. In the right administration pane, click the **Application Connections** tab.
3. Click **Register** to register a new in-bound application connection. The application is paired with a user and a device using the Activation Code supplied.
4. In Register Application Connection:
   a) Enter name of the user making the request from a particular device.
   b) Select the template to assign theapplication. The template you use supplies initial values in the subsequent fields.
      - Default – a default template that you can use as is, or customize.

- HWC – a default template for Hybrid Web Container. Use as is, or customize. If you use the HWC template, set the Application ID to HWC.
- Custom – select from a list of custom templates.

5. Change the default field values for the template you have chosen. If you are using the default template, you must provide the server name.
6. Click **OK**.

Index