SYBASE®
An SAP® Company

System Administration

Sybase Unwired Platform 2.1
ESD #1

DOCUMENT ID: DC01205-01-0211-01
LAST REVISED: December 2011

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

Contents

CHAPTER 1 Documentation Roadmap for Unwired Platform

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

See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role. *Fundamentals* is available on the Sybase Product Documentation Web site.

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

CHAPTER 2 Administration of the Unwired Platform

At its heart, Unwired Platform is a mobility-enablement platform. It has the tools, the client APIs, the server components and the administration console that offer a complete, end-to-end system for creating enterprise-level mobile solutions.

Administrators interact with Unwired Platform primarily to configure platform components and ensure the production environment works efficiently as a result of that configuration.

Unwired Platform delegates security checks, by passing login and password information to the security provider. In general, all administrative and application users and their passwords are managed in the security repository. Sybase Control Center limits feature visibility depending on the role an administrator logs in with. Unwired Platform administrators can be one of two types, each with distinct logins:

- Unwired Platform administrator (also known as the platform administrator) – has cluster-wide administration access to the Unwired Platform. supAdmin is the default login for cluster-side administration and is assigned the "SUP Administrator" role in PredefineUserLoginModule (the default Unwired Platform repository for development environments). In a deployment edition, you must map the SUP Administrator logical role to a role in your existing repository.
- Domain administrator – has access confined to the specific domains that the platform administrator assigns. supDomainAdmin is the default login for domain administration and is assigned the "SUP Domain Administrator" role in PredefineUserLoginModule. In a deployment edition, you must also map SUP Domain Administrator role to a role in your existing repository.

Both types of administrators are authenticated and authorized based on the 'admin' security configuration and its domain-level role mapping for administrative roles.

There are three main aspects to this platform that platform and domain administrators conjointly administer:

- Mobile application creation is supported by integrated development tools, APIs, samples and tutorials. For the developer, this aspect of the Unwired Platform allows the creation of integration logic and device applications that allow registered device users to seamlessly interact securely with your existing back-end infrastructure.

  Before you begin, know: the devices you need to support, the back-ends you need to integrate with, and the synchronization model you will use.
- Mobile application administration requires both the development and deployment of applications. The Unwired Platform perspective in Sybase Control Center is integral to configuring and managing applications as they are developed and integrated into your

system landscape. Further, all aspects of a production environment can be monitored for troubleshooting or performance tuning purposes.

Before you begin, decide: the system design/topology you environment requires and what type of security providers you need to delegate application security to.

- Mobile user, device, and application management simplifies how the end user is registered and provisioned in the Unwired Platform environment. When Afaria® is used in conjunction with Unwired Platform, an administrator has a powerful cross-platform mobile management framework:
  - Sybase Control Center performs the package deployment to the Unwired Server as well as manages user accounts and data service subscriptions.

  Before you begin, understand what package types you need to support, and how the package type affects how users are registered and subscriptions are created, and how your devices might be provisioned (cable or over-the-air).

Administration for Online Data Proxy

The Online Data Proxy runtime option for deploying OData SDK applications. Administration for Online Data Proxy requires a subset of administration information provided in this document. The information here guides you through the relevant sections and tasks for system administration.

- *Production Environment Administration*

  Production environment administration includes all the activities required to set up and configure your Unwired Platform server environment. These activities vary, depending on the license you have purchased. For example, personal systems and systems that support Online Data Proxy require fewer administration activities than what is required for an enterprise system in a production environment.
- *Device and Application Provisioning Overview*

  Provisioning describes all activities used to setup the device and its applications, so it can interact with the Unwired Platform environment.
- *Application and User Management*

  Application and user management applies to registering applications, creating application connections for users (with security configurations), and creating application connection templates).
- *System Logs*

  Unwired Platform uses multiple logs to record events that are useful for administrators who are monitoring the environment, and maintaining the health of components.
- *Operational Maintenance*

  Operational maintenance includes the regular administration activities that keep your mobile enterprise running properly.

- *Unwired Server Management API*

  Sybase Unwired Platform includes a Java API that opens the administration and configuration of Sybase Unwired Platform to Java client applications you create. By building a custom client with the administration client API, you can build custom application to support Sybase Unwired Platform administration features and functionality within an existing IT management infrastructure.
- *System Reference*

  To manage the system effectively, it is crucial to know about Unwired Platform subsystem components and how they fit together. This part outlines many important aspects of the system in quick reference format.

See also

- *Administration Areas Not Applicable to Online Data Proxy* on page 5

Administration Areas Not Applicable to Online Data Proxy

System Administration covers adminstration for all runtime options. If you are administering Online Data Proxy only, there are areas of administration that are not applicable.

Topics that are not applicable are identified with (Not applicable to Online Data Proxy) at the beginning of the topic.

In general, the following areas do not apply:

- Synchronization / Data Change Notification (DCN)
- SNMP Notifications
- Replication protocol
- Mobile Business Objects (MBOs), MBO packages
- Mobile Workflows or Hybrid Web Containers
- Package Administration
- Status and Performance Monitoring

See also

- *Administration for Online Data Proxy* on page 4

CHAPTER 3 Production Environment Administration

Production environment administration includes all the activities required to set up and configure your Unwired Platform server environment. These activities vary, depending on the license you have purchased. For example, personal systems and systems that support Online Data Proxy require fewer administration activities than what is required for an enterprise system in a production environment.

Cluster Administration Overview

The goal of cluster administration is to ensure that clusters and servers work smoothly, and scale over time. By default, the Unwired Platform is installed as a one-node cluster. The one-node cluster is supported in development or test environments. Production deployments of Unwired Platform are likely to require multiple nodes. Cluster administration is mostly a nonroutine administration task.

See *Installation Guide for Runtime > System Deployment Overview*.

Table 1. Cluster administration tasks

Task	Frequency	Accomplished by
Installing the cluster	One-time installation per cluster	Unwired Platform installer
Setting up relay servers	One-time initial installation and configuration; occasionally adding servers to the cluster	Manual installation; manual set-up using configuration files.
Suspending and resuming server nodes	On demand, as required	Sybase Control Center
Setting cluster properties, including cache database settings, monitoring database setup, and so on	Once, or as cluster changes require	Manual configuration using files and `.BAT` scripts.
Set up a shared data folder to share cluster synchronization content as an alternative method for keeping secondary nodes in sync.	Once, or as cluster changes require	Sybase Control Center

Task	Frequency	Accomplished by
Administering the runtime databases	Routine to ensure that the database server is monitored and backed up, that there is sufficient space for Unwired Platform metadata and cached data tables, and that performance is within acceptable limits (performance tuning)	Established processes and command line utilities. Consult with your database administrator.
Reviewing licensing information, including total licensed devices and currently used licenses count	Occasional, or as device user registration and deregistration occurs	Sybase Control Center.

Maintaining Host Names

Changing the host name is rarely done in a production system, but may be required as part of cluster administration, for example, if the host name is too long or includes reserved characters.

Changing Unwired Server Host Name

Change the Unwired Server host name. If you are changing the host name for a node in a cluster, the name must be changed for all nodes in the cluster. Additional steps are required if you are making a host name change after an upgrade.

1. Stop the Sybase Unwired Server service.
2. Update all the `socketlistener` property files related to the node with the changed host name. Change the value of host to the new host name. The property files are located at `${UnwiredServer}\Repository\Instance\com\sybase\djc\server\SocketListener\<mlservername>_<protocol>.properties`. Open each file, and change the old host value name to the new host name.
3. Update the property value in the clusterdb:
   a) Using dbisqlc, connect to the clusterdb.
   b) Update the sup.host property value with the new node value. For example:
   ```
   update MEMBER_PROP set value1='<new_hostname>' where
   name='sup.host' and value1='<old_hostname>'
   ```
4. Update the property value of sup.host with the new host name in `sup.properties`. It is located at: `Repository\Instance\com\sybase\sup\server\SUPServer\sup.properties`.
5. If you upgraded and then changed the Unwired Server host name, you need to complete additional steps (these steps are not needed for a fresh installation):

a) Change the listener prefix of httpListeners and iiopListeners for the new hostname in the new server's properties file:

```
Repository\Instance\com\sybase\djc\server\ApplicationServer
\default.properties,  <new_hostname>.properties
```

b) In `Repository\Instance\com\sybase\djc\server \SocketListner\*.properties`, rename all the `<old_hostname>_<protocol>.properties` into `<new_hostname>_<protocol>.properties`.

c) Use dbisql or dbisqlc to update the table: `cluster_installation in clusterdb, update cluster_installation set hostname='<new_hostname>' where hostname='<old_hostname>'`.

6. To enable Sybase Control Center to work:
   a) Restart the Sybase Control Center *X.X* service.
   b) Modify the URL in the shortcut for the SCC. Use the new hostname instead of the old one in the launched URL
7. For a cluster environment, repeat the same steps for each node.
8. Restart the Sybase Unwired Server service.

Configuring General Cluster Properties

Configure properties of a cluster to control whether or not a shared data folder is used, or whether asynchronous operation replays are enabled for all cluster packages.

1. In Sybase Control Center navigation pane, click the name of the cluster.
2. In the administration view, click **General**.
3. To synchronize data across Unwired Server nodes in the cluster:

   In a production environment, Sybase recommends using a shared data folder to manage communications between the primary node in the cluster, and secondary nodes. For details, see *Shared Data Folders* .

   a) To enable cluster synchronization, click **Cluster sync data shared path enabled**.
   b) Configure the shared data path.

      In **Cluster sync shared data path** set the shared folder using the UNC convention (`\\hostname\shared_folder_name`). The shared folder must have write access.
4. Configure the queue limit for asynchronous operation replays in **Asynchronous operation replay queue count**. The minimum acceptable queue count is 1 and the default is 5.

   The queue for operation replays can then be viewed by a package. See *Viewing Asynchronous Operation Replays* in Sybase Control Center online help.

Shared Data Folders

By default, Unwired Server secondary nodes in a cluster use an internal interface to retrieve content from the primary node. In a production environment, it may be more appropriate to use a shared data folder to share the cluster synchronization content.

The shared data folder includes the data tier database and transaction log files. This enables you to set up the information once, and share with all nodes.

Couple this with a highly available configuration, where processing will failover and to a secondary node that becomes the primary in certain circumstances, all nodes have access to the most current information at all times.

You typically set up the shared data folder as part of the installation process, but you can add the file later if you set up or reconfigure a cluster after installation. One directory on a shared cluster storage must be designated as the shared data folder. It must be shared for network access, either as a file share resource group, or as a Client Access Point.Keep in mind these guidelines:

- Each node in the Unwired Server cluster must be able to access the shared data folder, from outside the failover cluster.
- Unwired Server processes on each Unwired Server host must have read/write permission in the shared data folder.
- The shared data folder can be a parent directory, common to all database files and transaction logs, but it does not have to be.

Server Administration Overview

The goal of server administration is to ensure that Unwired Server is running correctly and that it is configured correctly for the environment in which it is installed (development or production). Server administration is mostly a one-time or infrequent administration task.

Table 2. Server administration tasks

Task	Frequency	Accomplished by
Installing the server	One-time installation per server	Unwired Platform installer.

Task	Frequency	Accomplished by
Configuring the server to: • Set the replication and messaging synchronization ports, as well as communication ports for administration and DCN • Create security profiles for secure communication • Set up secure synchronization • Configure replication and messaging push notifications • Tune server performance	Postinstallation configuration with infrequent tuning as required	Sybase Control Center for Unwired Platform. See Sybase Unwired Server in Sybase Control Center online help
Manage the outbound enabler configuration for Relay Server. • Configure Relay Server properties • Manage certificates • View logs • Configure proxy servers for outbound enabler	Postinstallation	Sybase Control Center for Unwired Platform.
Setting server log file settings and subsystem log levels	Once, unless log data requirements change	Sybase Control Center for Unwired Platform. See *Server Log* in Sybase Control Center online help.

General Server

Configure properties and security profiles for Unwired Server management and communication ports. These ports process incoming replication synchronization, administration, and data change notification requests. You must secure data transmission over management and DCN communication ports by creating and assigning an SSL configuration to the ports. You can also configure Unwired Server performance properties.

See *General Server* topics in *Sybase Control Center* online help.

Configuring Unwired Server Performance Properties

To optimize Unwired Platform performance, configure the thread stack size, maximum and minimum heap sizes, user options, and inbound and outbound messaging queue counts.

1. Open Sybase Control Center.
2. In the left navigation pane, expand the **Servers** folder and select a server.
3. Select **Server Configuration**.

4. In the right administration pane, select the **General** tab.
5. From the menu bar, select **Performance Configuration**.
6. Configure these properties, as required:
   - Host Name – the name of the machine where Unwired Server is running (read only).
   - Thread Stack Size – the JVM `-Xss` option.
   - Minimum Heap Size – the minimum size of the JVM memory allocation pool, in megabytes. For production recommendations on this value, see *Unwired Server Replication Tuning Reference* in *System Administration*.
   - Maximum Heap Size – the maximum size of the JVM memory allocation pool, in megabytes. FFor production recommendations on this value, see *Unwired Server Replication Tuning Reference* in *System Administration*.
7. (Optional) Expand the **Show optional properties** section and configure these properties, as required:
   - Maximum Number of In Memory Messages – specify the number of in-memory messages to allow.
   - User Options – other JVM options. For example, you can enable JVM garbage collection logging by setting `-XX:+PrintGCDetails`. Or you can set the permanent space which allocated outside of the Java heap with `DJC_JVM_MAXPERM`; the maximum perm size must be followed by K, M, or G, for example, `-XX:MaxPermSize=512M`. Note that DJC_JVM_MAXPERM is not visible to Sybase Control Center.
   - Inbound Messaging Queue Count – the number of message queues used for incoming messages from the messaging-based synchronization application to the server. Sybase recommends a choose a value that represents at least 10% of active devices.
   - Outbound Messaging Queue Count – the number of message queues used for outbound messages from the server to the messaging-based synchronization application. Sybase recommends a choose a value that represents at least 50% of active devices. However, if you are running 32-bit operating system, do not exceed a value of 100% of active devices.
   - Subscribe Bulk Load Thread Pool Size – the maximum number of threads allocated to initial bulk load subscription operations. The default value is five. Setting the thread pool size too high can impact performance.

   Note: If you increase either queue count property, ensure you also increase the MaxThread property in the `<hostname>_iiop1.properties` file.
8. Click **Save**.

Applying Performance Tuning Changes if Unwired Server is a Service

Certain Unwired Server tuning changes require additional steps to apply the changes.

Follow these recommendations after changing the configuration of certain properties.

For the CDB pool size, no additional action is required. For all other settings, a server restart is required for those settings to take effect.

Messaging Content Sizes

A new property was added to support messaging performance. The message size limitation can be changed by modifying an internal Unwired Server property, `sup.msg.max_content_size`. If you feel the 20KB value should be changed, work with your Sybase representative.

The current message size limit for Unwired Server is 20KB. In general, enlarging the message size results in a lower number of messages, and higher efficiency.

Performance also depends on the device environment. A message that is too large stresses the device, and negates efficiency. Device factors include memory and size of the object graph being sent. In some cases, a larger message size terminates message processing. When the Unwired Server message size exceeds the limit, the message is immediately sent to the client side.

Methods for Starting and Stopping Unwired Server

You can start and stop Unwired Server depending on the use context. Review this table to understand which method you should use.

Method	Use when	Services stopped
Sybase Control Center Unwired Server list	Stopping and starting remote Unwired Server nodes.	Unwired Server service only.
Desktop shortcut	Stopping Unwired Server locally.	All runtime services installed on that host.
Windows Services	Stopping Unwired Server locally.	Any combination of individual services that require stopping.

Stopping and Starting a Server from Sybase Control Center

Stop and start a server to perform maintenance or to apply changes to server settings. You can perform this action as a two-step process (stop and start) or as a single restart process.

You can stop and start a server from Sybase Control Center for servers that are installed on the same host as Sybase Control Center, as well as servers that are installed on different hosts.

Note: If someone manually shuts the server down, this action triggers multiple errors in Sybase Control Center for Unwired Server until the console determines that the server is no longer available. This takes approximately 30 seconds to detect. When this occurs you might see multiple `Runtime API throws exception` errors logged. Wait for the server to come online and log into the server again to resume your administration work.

1. In the Sybase Control Center navigation pane, click **Servers** to display the servers list.

2. Select one or more servers in this list.
3. Choose an appropriate process:
   - To stop the server, click **Stop**. You can then perform the administration actions you require that might require the server to be started. To then restart the server, click **Start**.
   - If you perform an administration action that requires a restart to take effect, click **Restart**. This shuts the server down and restarts it in a single process.

As the server stops and starts, progress messages display in the Server Console pane.

Stopping and Starting a Server from the Desktop Shortcut

If stopping all local Unwired Platform runtime services in addition to the Unwired Server, you can use the desktop shortcut installed with Unwired Platform runtime components

Only use this method if you want all local services installed on the host stopped or started. Otherwise choose a different start or stop method.

1. To stop Unwired Server and related runtime services, click **Stop Sybase Unwired Platform Services** from the host's desktop.
2. To start Unwired Server and related runtime services, click **Start Sybase Unwired Platform Services** from the host's desktop.

Replication

Replication synchronization involves synchronization between Unwired Server and a replication-based mobile device application. Synchronization keeps multiple variations of the data set used by a device application in coherence with one another by reconciling differences in each. Reconciling differences before writing updates back to the enterprise information server (EIS) maintains data integrity.

For replication synchronization, configure the corresponding port to receive incoming synchronization requests from devices, as well as set up configuration to enable push notification messages to the device when data changes in CDB. In a typical environment, client applications running on devices will connect to the synchronization port via Relay Server and Relay Server Outbound Enabler (RSOE). In those cases, the HTTP port will be used.

See *Replication* topics in *Sybase Control Center* online help.

Configuring a Replication Listener

(Not applicable to Online Data Proxy)Configure the port to receive synchronization requests from client devices. If you are using push synchronization, then also configure synchronization listener properties.

Prerequisites

A secure synchronization stream uses SSL or TLS encryption. Both TLS and SSL require production-ready certificates to replace the default ones installed with Unwired Platform.

Ensure that you possess digital certificates verified and signed by third-party trusted authorities. See *Encrypting Synchronization for Replication Payloads* in *Security*.

Task

1. Open Sybase Control Center.
2. In the left navigation pane, expand the **Servers** folder and select the primary server.

   Note: You can only configure the encryption properties on the primary Unwired Server. Secondary servers will inherit the values, where they become read-only.
3. Select **Server Configuration**.
4. In the right administration pane, click the **Replication** tab.
5. If push synchronization is being added to the application, select **Listener** from the menu bar.
6. Select the protocol and port you require:

   - If you do not require SSL encryption, choose **Port**. Sybase recommends this option if you do not require a secure communication stream for synchronization. By default, the port for HTTP is 2480.
   - To encrypt the HTTP stream with SSL, choose **Secure port**. By default, the port for HTTPS is 2481. The "Secure Sync Port" properties in the "Optional Properties" table can be used to review and set the server identity and public certificate for the secure synchronization port. See below.
7. Configure these properties:

   - **Synchronization Cache Size** – sets the maximum cache size for the synchronization port. The default is 50MB.
   - **Thread Count** – sets the number of worker threads used for synchronization. The default is 10. If you experience performance issues, you may need to increase this value.
8. (Optional) Expand the optional properties section to configure additional properties for E2EE with TLS, HTTPS with SSL, and synchronization server startup options:

   Note: Leave E2E Encryption values blank to disable end-to-end encryption.

   - E2E Encryption Certificate Password – set the password to unlock the encryption certificate.
   - E2E Encryption Certificate – specify the file containing the private key that acts as the identity file for Unwired Server.
   - E2E Encryption Type – specify the asymmetric cipher used for key exchange for end-to-end encryption. You can only use RSA encryption.
   - Secure Sync Port Certificate – identifies the location of the security certificate used to encrypt and decrypt data transferred using SSL.

- Secure Sync Port Certificate Password – is used to decrypt the private certificate listed in certificate file. You specify this password when you create the server certificate for SSL.
- Secure Sync Port Public Certificate – specify the file containing the SSL public key that acts as the identity file for synchronization port.
- Trusted Relay Server Certificate – if Relay Server trusted certificate is configured for HTTPS connections encrypted with SSL, identifies the public security certificate location.
- User Options – sets the command line options for starting the synchronization server. These options are appended the next time the synchronization server starts. These are the available user options:

Option	Description
`@ [variable | filePath]`	Applies listener options from the specified environment variable or text file.
`-a <value>`	Specifies a single library option for a listening library.
`-d <filePath>`	Specifies a listening library.
`-e <deviceName>`	Specifies the device name.
`-f <string>`	Specifies extra information about the device.
`-gi <seconds>`	Specifies the IP tracker polling interval.
`-i <seconds>`	Specifies the polling interval for SMTP connections.
`-l <"keyword=value;...">`	Defines and creates a message handler.
`-m`	Turns on message logging.
`-ni`	Disables IP tracking.
`-ns`	Disables SMS listening.
`-nu`	Disables UDP listening.
`-o <filePath>`	Logs output to a file. **Note:** Ensure that you enter the absolute file path for this property.
`-os <bytes>`	Specifies the maximum size of the log file.
`-p`	Allows the device to shut down automatically when idle.

Option	Description
`-pc [+ | -]`	Enables or disables persistent connections.
`-r <filePath>`	Identifies a remote database involved in the responding action of a message filter.
`-sv <scriptVersion>`	Specifies a script version used for authentication.
`-t [+ | -] <name>`	Registers or unregisters the remote ID for a remote database.
`-u <userName>`	Specifies a synchronization server user name.
`-v [0 | 1 | 2 | 3]`	Specifies the verbosity level for the messaging log.
`-y <newPassword>`	Specifies a new synchronization server password.

Do not use the User Options property in Sybase Control Center to pass in these options: `-c`, `-lsc`, `-q`, `-w`, `-x`, `-zs`.

For more information on synchronization server command line options, see *MobiLink Listener options for Windows devices* (*http://infocenter.sybase.com/help/topic/com.sybase.help.sqlanywhere.12.0.1/mlsisync/ms-listener-s-3217696.html*) in the *SQL Anywhere® 12.0.1* online help.

9. Click **Save**.

Messaging

Messaging is a synchronization method used to maintain data integrity on device client applications. It uses a JMS service to upload and download data changes to and from the Unwired Server cache database. Messaging-based synchronization ports implement a strongly encrypted HTTP-based protocol using a proprietary method.

Configure messaging in the Messaging tab of the Server Configuration node for the particular server you are administering.

See *Messaging* topics in *Sybase Control Center* online help.

Configuring Messaging Synchronization Properties

(Not applicable to Online Data Proxy) Configure one or more synchronization ports to receive service requests from devices.

1. Open Sybase Control Center.
2. In the left navigation pane, expand the **Servers** folder and select a server.
3. Select **Server Configuration**.
4. In the right administration pane, click the **Messaging** tab, and select **Listener**.

5. Enter the synchronization port number. The default is 5001.
6. (Optional) Select **Listen on multiple ports** and enter the additional port numbers.

   Depending on your environment, listening on multiple synchronization ports may provide greater flexibility and reliability. High activity on particular ports, such as virus detection and data inspection, may result in dropped packets or connections if alternate ports are unavailable. When multiple ports are configured, all messaging traffic is still funneled to a single listener.
7. Click **Save**.

See also

- *Configuring Apple Push Notification Service* on page 55
- *Configuring Push Notifications for the BlackBerry Enterprise Server* on page 59

Relay Server Outbound Enabler

The Outbound Enabler (RSOE) runs as an Unwired Server process and manages communication between the Unwired Server and a Relay Server.

Each RSOE maintains connections to each Relay Server in a Relay Server farm. The RSOE passes client requests to the Unwired Server on its Replication or Messaging port. Unwired Server sends its response to the RSOE, which forwards it to the Relay Server, to be passed to the client.

As an Unwired Server process, the RSOE always starts when Unwired Server starts. Unwired Server monitors the process to ensure it is available. If an RSOE fails for any reason, Unwired Server restarts it automatically.

Note: Sybase recommends three RSOE processes each, for both Replication and Messaging ports.

See *Relay Server Outbound Enabler* topics in *Sybase Control Center* online help.

Data Tier Administration Overview

If you need to change properties of the data tier, review the tasks you can perform for different Unwired Platform databases.

If you need to change the password for the DBA user, see either *Changing DBA Passwords for SQLAnywhere Databases in a Single Node Installation* or *Changing DBA Passwords for SQLAnywhere Databases in a Cluster Deployment* in the *Security* guide.

Changing Database Ports for SQLAnywhere Databases

By default, ports are configured when you install the data tier. You can change values for any SQLAnywhere database deployed as part of the data tier.

During installation the you are prompted to enter a port number for each of the SQLAnywhere databases used by the runtime: the cache database (CDB), the cluster database, the log database, and monitor database.

Depending on whether or not you have deployed Unwired Platform as a single node (as in the case of Online Data Proxy environments), or as a cluster, the process varies slightly. This is because:

- In single node deployment, a single database server named CacheDB supports all installed databases.
- In a cluster deployment, two servers are used: the monitor and domain log databases use a server called LogDataDB, the default database used for the cache data uses the CacheDB server, and the cluster database uses the ClusterDB server.

1. Stop all instances of Unwired Server, as well as all database services.

   For a list of database services, search for *Unwired Platform Windows Services* in *System Administration*.
2. If you are updating ports in a cluster deployment, then for each database that requires a port change, open the corresponding initialization file. These files are installed to `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere12\binxx`:

   - For the cache database, open `cdboptions.ini`.
   - For the cluster database, open `cldboptions.ini`.
   - For the log database, open `monitoroptions.ini`.
3. For all deployments, run the update properties utility to propagate changes to all runtime servers:

   ```
   updateProps.bat -u user -p pwd -d dsn -nv
   "serverport_property1=newport#serverport_property2=newport
   ```

   Supported server port property names include:

   - cdb.serverport
   - cldb.serverport
   - monitoringdb.serverport
   - domainlogdb.serverport

   For details on the update properties utility, see *Update Properties (updateprops.bat) Utility* in the *System Administration* guide.
4. Restart all database services, then all Unwired Servers.

See also

- *Changing SQLAnywhere Database Server Startup Options* on page 20
- *Update Properties (updateprops.bat) Utility* on page 250

Changing SQLAnywhere Database Server Startup Options

Change the database server startup options to control the performance of a SQLAnywhere database server.

Depending on the type of installation you have performed, the servers you can change startup options for varies:

- In single node deployment, a single database server named CacheDB supports all installed databases.
- In a cluster deployment, two servers are used: the monitor and domain log databases use a server called LogDataDB, the default database used for the cache data uses the CacheDB server, and the cluster database uses the ClusterDB server.

1. Stop all instances of Unwired Server, as well as all database services.

   For a list of database services, search for *Unwired Platform Windows Services* in *System Administration*.
2. If you are updating ports in a cluster deployment, then for each database that requires a port change, open the corresponding initialization file. These files are installed to `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere12\bin`*xx*:
   - For the cache database, open `cdboptions.ini`.
   - For the cluster database, open `cldboptions.ini`.
   - For the log database, open `monitoroptions.ini`.
3. Modify one of these properties: *.threadcount or *.useroptions in the `.ini` files, for example:

   ```
   -c 24M -gn 300
   ```
4. For all deployments, run the update properties utility to propagate changes to all runtime servers:

   ```
   updateProps.bat -u user -p pwd -d dsn -nv
   "property1=newvalue#property2=newvalue
   ```
5. Restart all database services, then all Unwired Servers.

See also

- *Changing Database Ports for SQLAnywhere Databases* on page 19
- *Update Properties (updateprops.bat) Utility* on page 250

Changing the Cache Database Server Thread Count

You can change the cache database (CDB) server thread count as required. The procedure for this varies depending on the type of environment the CDB server is deployed to.

Sybase has identified two scenarios: one for clustered environments and one for single node installations.

Note: You can set these values for monitoring and cluster databases as well.

See also

- *Update Properties (updateprops.bat) Utility* on page 250

When data-tier in a cluster environment

1. Stop all runtime nodes in the cluster.
2. On the data-tier node, remove the service by running:

   ```
   <UnwiredPlatform_InstallDir>\Servers\SQLAnywhereXX\data
   \install-sup-sqlanyXX.bat remove
   ```
3. To change the thread count, edit the `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhereXX\data\install-sup-sqlanyXX.bat` to modify the CDB_THREADCOUNT property.

   Choose an appropriate server threadcount according to your cluster performance requirements.

   For example, this command sets the server thread count to 200:

   ```
   set CDB_THREADCOUNT=200
   ```
4. Recreate the service again with the correct thread count by running:

   ```
   <UnwiredPlatform_InstallDir>\Servers\SQLAnywhereXX\data
   \install-sup-sqlanyXX.bat install [auto|manual]
   cdb_servername cdb_serverhost cdb_serverport cdb_password
   cldb_serverhost cldb_serverport cldb_password
   monitordb_serverhost monitordb_serverport
   monitordb_password domainlogdb_password [shared_data_path
   service_username service_password]
   ```
5. Restart all runtime nodes in the cluster.

When data-tier is in a personal or evaluation install

1. On the Unwired server node with data-tier installed, update properties by running:

   ```
   <UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin
   \updateProps.bat -nv "cdb.threadcount=200"
   ```

Choose an appropriate server threadcount according to your cluster performance requirements.

2. Stop Unwired Server services.
3. Start Unwired Server services.

Changing the Cache Database Server Pool Size

You can change the cache database (CDB) server maximum pool (and other configuration values) through Sybase Control Center. In a cluster environment, the CDB pools size can propagate to secondary notes.

1. In the left navigation pane, expand the **Domains** folder, and select the domain for which you want to modify the connection.
2. Select **Connections**.
3. In the right administration pane:
   - To edit the properties of a connection pool, click the **Connections** tab.
   - To edit the properties of a connection pool template, click the **Templates** tab.
4. Select a connection pool or template from the list.
5. Select **JDBC** as the **Connection pool type**, and click **Properties**.
6. Change the Max Pool Size value (and any other values you choose). The default value is 150, and 0 indicates no limit. Th Max Pool Size value should not be a negative value.

   See *Creating Connections and Connection Templates* in *Sybase Control Center* online help, and *JDBC Properties*.
7. Click **Save** to save the changes.

See also

- *JDBC Properties* on page 207

Using a Different Database Log Path

Sybase recommends that you always use the default database log path location for Unwired Platform databases. However, the database log path can be changed.

Note: This information is useful if you want database and log files on separate IO channels for better performance, or other reasons, after installation.

1. Change directories to `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhereXX\BINXX`.
2. To move a log from its default location to a new drive, use a command similar to this one:

   ```
   dblog -t <New Log File Name> -m <Mirror Log File Name>
   <Database file Name>
   ```

   For example:

```
dblog -t D:\databaseLog\default.log -m E:\databaseLog
\default_mirror.log
C:\databases\default.db
```

This example, moves `default.log` from its default location of `C:\databases`, to the `D:\` drive, and sets up a mirrored copy on the `E:\` drive. Sybase recommends naming the log files specified by -t and -m options differently.

3. You can use this same syntax for other databases, like monitoring database, simply by changing the log and database file names appropriately.

See also

- *Sample Backup and Recovery Plan* on page 172

Cache Database and Timezones

All nodes in a cluster must run with the same timezone setting.

Setting Up an Existing Database for Monitoring

You can use any SQL database provided that the existing version number of that database matches the version required by Unwired Platform.

Setting up the existing database requires some changes to the schema. Once setup, you can configure Unwired Platform to use this database.

1. Use dbisql to run the SQL scripts that set up the monitoring schema. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhereXX\BINXX`.
   - `init-monitoring-tables-asa.sql` – sets up the SQL Anywhere database with a general monitoring schema.
   - `ASA_MONITORING_DDL.sql` – sets up the SQL Anywhere database with a cache monitoring schema.

   By default, these scripts are installed in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\config`.
2. Create a new data source for the monitoring database in the **default** domain.
3. In Sybase Control Center, configure Unwired Server to use this database instance. See *Configuring Monitoring Performance Properties* in Sybase Control Center online help.
4. Configure monitoring behavior accordingly.

Isolating the Monitoring and Domain Logging Databases from Cache and Messaging Databases

Install a new data tier node and configure the connection details to it.

1. Perform a data-tier only install on a new data tier node.

This installs the monitor and domain log database as a service (named Sybase Unwired LogData Service).

2. Disable the cache, cluster and messaging database services.
3. Connect to Unwired Server cluster from Sybase Control Center and:
   a) In the navigation pane, expand the **default** domain node, then click **Connections**.
   b) Click the **Connections** tab.
   c) For each connection (monitordb and domainlogdb), click **Properties**, then update the host and port values to reflect the connection properties of the new host node.
   d) Click Save.

See also

- *Planning for Domain Logging* on page 151

EIS Connection Management Overview

The goal of enterprise information system connection management is to ensure the connections to back-end repositories of data remain available to Unwired Server and deployed packages that require those connections. Connections management is a non-routine administration task.

EIS connections include:

- JDBC
- SAP® Java Connector
- SAP DOE-C
- Proxy
- Web Service

Review the tasks outlined in this table to understand the data management workflow for each role and the degree of activity it entails.

Task	Frequency	Accomplished by
Create and tune connections	On-demand as needed	Sybase Control Center for Unwired Platform with the Connections node
Create and maintain connection pool templates	One-time	Sybase Control Center for Unwired Platform with the Connections node

Task	Frequency	Accomplished by
Update package connections when moving from development and test environments to production environments	On-demand, as needed	Sybase Control Center for Unwired Platform with the **Deployment Wizard**

See also

- *EIS Data Source Connection Properties Reference* on page 206
- *JDBC Properties* on page 207
- *SAP Java Connector Properties* on page 219
- *SAP DOE-C Properties* on page 224
- *Proxy Properties* on page 226
- *Web Services Properties* on page 227
- *Creating Data Source Connections for a Domain* on page 32

Data Source Connections

A data source connection is a physical connection definition that provides runtime connection to enterprise information systems (EIS), that in turn enables data to be mobilized by Unwired Server to client device via synchronization or messaging. Before you create publications or subscriptions, or deploy packages, you must first define database connections.

For Unwired Server to recognize a EIS data source, you must define a connection to that data repository. The connections are defined in Sybase Control Center with the Unwired Platform perspective, and are known as server-to-server connections because they are opened by Unwired Server.

In Unwired Platform you create a connection template from which you can replicate connections. Connection pools allows Unwired Servers to share pools of pre-allocate connections to a remote EIS server. This preallocation avoids the overhead imposed when each instance of a component creates a separate connection. Connection pooling is only supported for database connections, and the size of the pool is controlled by the Max Pool Size property of the connection.

Connection Templates

A connection template is a model or pattern used to standardize connection properties and values for a specific connection pool type so that they can be reused. A template allows you to quickly create actual connections.

Often, setting up a connection for various enterprise data sources requires each administrator to be aware of the mandatory property names and values for connecting to data sources. Once you create a template and add appropriate property names and corresponding values (for example user, password, database name, server name, and so on), you can use the template to instantiate actual connection pools with predefined property name and value pairs.

Changing Connections to Production Data Sources

Platform and domain administrators can change endpoint connection information when moving applications from development or test environments to production environments.

Review this list to determine what connection elements may be affected when transitioning between these different environments:

1. You can change endpoint connection mapping when the mobile business objects (MBOs) are deployed to the production Unwired Server.

   Make these changes using either Unwired WorkSpace development tools for design-time changes, or Sybase Control Center for Unwired Platform for deployment-time changes.
2. You need not change MBOs, if the developer uses production data source connection credentials. MBOs that use user name and password personalization keys to authenticate server connections, do not use the user name and password specified in the server connection.

   The client user credentials are used to establish connection with the back-end. Administrators should determine from their development team which MBOs are affected. You must still remap connections to production values.
3. (Optional) Tune properties (such as connection pool size), to improve the performance of production-ready applications.

Viewing and Editing EIS Connection Properties

View these settings when troubleshooting connection problems, and make changes as needed to correct a problem or improve performance.

1. Log in to Sybase Contol Center.
2. Expand the domain and select **Connections**.
3. On the **Connections** tab, in the **Connection Pool Name** list, select the connection pool.
4. Click the **Properties** button to display the current settings for the connection pool.
5. See the reference topic linked below for detailed information about the different settings.
6. If you change any settings, use the **Test Connection** button to ensure that your changes work before saving them.

See also

- *EIS Data Source Connection Properties Reference* on page 206

Domain Administration Overview

The goal of domain management is to create and manage domains for one specific tenant. Use multiple domains for multiple tenants sharing the same Unwired Server cluster.

Multiple domains in a cluster allow tenants' administrators (that is, domain administrators) to each manage their own application components. Domain administration for the platform administrator is typically an infrequent administration task that occurs each time a new domain needs to be added to support a change in the tenancy strategy used or need to make changes to an existing domain.

Domains give you the means to logically partitioning environments, thereby providing increased flexibility and granularity of control over domain-specific applications. Administration of multiple customer domains takes place within the same Unwired Platform cluster.

- An Unwired Platform administrator adds and configures domains, creates security configurations for customer applications, and assigns those security configurations to the domain so they can be mapped to packages in the domain.
- One or more domain administrators then perform domain-level actions within their assigned domains.

In a development environment, domains allow developers from different teams to share a single Unwired Server cluster without disrupting application deployment. Administrators can facilitate this by:

1. Creating a domain for each developer or developer group.
2. Granting domain administration privileges to those users so they can perform deployment tasks within their assigned domains.

Table 3. Domain management tasks

Task	Frequency	Administrator
Create domains	Once for each customer	Unwired Platform administrator
Create and assign security configurations, and map roles at package or domain levels	Infrequent, as required	Unwired Platform administrator
Assign and unassign domain administrators	Infrequent, as required	Unwired Platform administrator
Configure and review domain logs	Routine	Unwired Platform administrator and domain administrator

Task	Frequency	Administrator
Deploy MBO and DOE-C packages	Routine	Unwired Platform administrator and domain administrator
Manage server connections and templates	Infrequent, as required	Unwired Platform administrator and domain administrator
Manage subscriptions and scheduled tasks	As required	Unwired Platform administrator and domain administrator
Review client log and MBO/operation error history	As required	Unwired Platform administrator and domain administrator

Enabling a Multitenancy Environment with Domains

Platform administrators can add new domains to the Unwired Platform environment to facilitate tenants' administration of their own components.

By default, Unwired Platform uses the Default domain. A single domain does not offer a logical partioning of the mobility environment, which is crucial if you need to support multiple tenants. The number of domains you need to add is determined by the strategy you employ.

Once the setup is complete, domain administrators can manage domain artifacts.

1. *Determining a Tenancy Strategy*

   Determine how many domains to create and how to distribute domain components. A strategic multitenant structure for the cluster balances system-availability considerations with administrative concerns.
2. *Creating and Enabling a New Domain*

   Create and configure multiple domains within a single Unwired Platform installation. A domain must be enabled for application users to access the packages deployed in the domain. Enabling a domain also triggers synchronization of the domain changes to the secondary nodes in the cluster. Application users who attempt to access a disabled domain receive an error message.
3. *Creating a Security Configuration for a Domain*

   Define a set of security providers in Sybase Control Center to protect domain resources, according to the specific security requirements of the domain. A security configuration can either be created first and then mapped to the desired domain.
4. *Activating a Domain Administrator*

A platform administrator must create and register domain administrators, before this individual can access a domain.

5. *Assigning Domain Administrators to a Domain*

   Assign domain administration privileges to a domain administrator. You must be a platform administrator to assign and unassign domain administrators.

6. *Mapping Roles for a Domain*

   Configure role mapping in Sybase Control Center to manage logical roles and authorize client requests to access domain resources.

7. *Creating Data Source Connections for a Domain*

   A connection is required to send queries to mobile business objects and receive data. Configure the properties required to connect to EIS datasources.

Determining a Tenancy Strategy

Determine how many domains to create and how to distribute domain components. A strategic multitenant structure for the cluster balances system-availability considerations with administrative concerns.

Domains are primarly containers for packages for a specific group. This group is called a tenant and can be internal to a single organization or external (for example, a hosted mobility environment).

Packages are attached to named security configurations that determine which users have access to mobile business object data, so you must create at least one security configuration and assign it to the domain for which the package is being deployed. You must identify which users require access to each package and how you will distribute the packages across the system using domains to logically partition the environment.

1. Organize device users according to the data they need to access. Ideally, create a domain for each distinct set of users who access the same applications and authenticate against the same back-end security systems. If you do not need to support multiple groups in distinct partitions, then the single default domain should suffice.
2. Consider how these groups will be affected by administrative operations that prevent them from synchronizing data. Sometimes, you can limit the number of users affected by administration and maintenance disruptions by distributing packages across additional domains. Operationally, the more components a domain contains, the more clients who are unable to access package data during administrative operations like domain synchronizations.
3. Assess the administrative resources of the tenant to determine how much time can be committed to domain administration tasks. Certain multitenant configurations require a greater amount of administrative time. For example, if you distribute packages from the same EIS across a number of domains, you must maintain identical data source configurations for each of these packages. This option requires more administrative time than grouping all packages belonging to the same EIS into one domain.

4. Decide how many domains to create for the customer, and identify which packages to group within each domain, according to the needs of the user groups you identified in step 1.

Creating and Enabling a New Domain

Create and configure multiple domains within a single Unwired Platform installation. A domain must be enabled for application users to access the packages deployed in the domain. Enabling a domain also triggers synchronization of the domain changes to the secondary nodes in the cluster. Application users who attempt to access a disabled domain receive an error message.

Prerequisites

Create a security configuration for the domain and register the domain administrator.

Task

1. Open Sybase Control Center.
2. In the left navigation pane, select the **Domains** folder.
3. In the right administration pane, select the **General** tab, and click **New**.
4. In the Create Domain dialog, enter a name for the domain and click **Next**.
5. Optional. Select a security configuration for the domain by checking an option from the list of available configurations. These security configurations are then available for use in validating users accessing the packages.
6. Click **Next**.
7. Optional. Select one or more domain administrators for the domain.
8. Click **Finish**.
   The new domain appears in the **General** tab.
9. Click the box adjacent to the domain name, click **Enable**, then click **Yes** to confirm.

Creating a Security Configuration for a Domain

Define a set of security providers in Sybase Control Center to protect domain resources, according to the specific security requirements of the domain. A security configuration can either be created first and then mapped to the desired domain.

A security configuration determines the scope of data access and security. A user must be part of the security repository used by the configured security providers to access any resources (that is, either a Sybase Control Center administration feature or a data set from a back-end data source) on a domain. See *Security Configurations* in the *Security* guide.

1. In Sybase Control Center, add a new security configuration using the **Security** node.
2. In the left navigation pane, expand the **Security** folder and select the new configuration.

3. Use the **Authentication**, **Authorization**, and **Audit** tabs to configure the appropriate security providers for each aspect of domain security.
4. Edit the security provider properties, as required.
5. Validate the configuration to ensure that Unwired Server accepts the changes.
6. Apply the changes to Unwired Server.

Activating a Domain Administrator

A platform administrator must create and register domain administrators, before this individual can access a domain.

Prerequisites

Domain administrator required physical roles of the security repository should already be mapped to the default SUP Domain Administrator logical role in 'admin' security configuration in 'default' domain.

Task

1. In the Security node of Sybase Control Center, create a new domain administrator user by providing the: login, company name, first name, and last name.

   See *Registering a Domain Administrator User* in Sybase Control Center help.
2. Assign the login the required physical role in the security provider repository to which the 'admin' security configuration is pointing. This is accomplished by using the tool provided by the security provider. See *Authorization* in the *Security* guide.

Assigning Domain Administrators to a Domain

Assign domain administration privileges to a domain administrator. You must be a platform administrator to assign and unassign domain administrators.

Prerequisites

Ensure the user is already registered as a domain administrator in the Domain Administrators tab.

Task

1. Open Sybase Control Center.
2. In the left navigation pane, expand the **Domains** folder, and select the domain for which to assign domain administration privileges.
3. Select the domain-level **Security** folder.
4. In the right administration pane, select the **Domain Administrators** tab, and click **Assign**.

5. Select one or more administrator users to assign to the domain by checking the box adjacent to the user name.
6. Click **OK**.
   A message appears above the right administration pane menu indicating the success or failure of the assignment. If successful, the new domain administrator appears in the list of users.

See also

- *Mapping Roles for a Domain* on page 79

Creating Data Source Connections for a Domain

A connection is required to send queries to mobile business objects and receive data. Configure the properties required to connect to EIS datasources.

The format in which data is communicated depends on the type of datasource. Establish connections by supplying an underlying driver and a connection string that allow you to address the datasource, and provide you a mechanism by which to set the appropriate user authentication credentials and connection properties. See *EIS Connection Management Overview*, and *Connections* in *Sybase Control Center* online help.

Note: When creating connections, please ensure that the prerequisites for each connection type are installed on all nodes of the cluster.

1. Open Sybase Control Center.
2. Select the domain-level **Connections** node for the domain to configure.
3. Create a new connection or connection template by selecting the appropriate tab and clicking **New**.
4. Enter a unique connection pool name, and select both a connection pool type and the appropriate template to use for that type. Customize the template, if required, by editing existing values or adding new properties.
5. Test the values you have configured by clicking **Test Connection**. If the test fails, either the values you have configured are incorrect, or the datasource target is unavailable. Evaluate both possibilities and try again.
6. Click **OK** to register the connection pool.
   The name appears in the available connection pools table on the Connections tab; administrators can now use the connection pool to deploy packages.

See also

- *EIS Connection Management Overview* on page 24
- *Mapping Roles for a Domain* on page 79

Registering Sybase Unwired Platform with System Landscape Directory

The System Landscape Directory (SLD) is a central repository of system landscape information used to manage the software lifecycle. The SLD describes the systems and software components that are currently installed. SLD data suppliers register the systems on the SLD server, and keep the information up-to-date. Sybase Unwired Platform is a third-party system that must be registered with the SLD.

Prerequisites

- An SLD of SAP NetWeaver 2004s SPS07 or higher is installed.
- The SLD server is running.
- SLD has been configured to receive data. For more information, see the *Post-Installation Guide* and the *User Manual* for your SAP NetWeaver version on SDN: *http://www.sdn.sap.com/irj/sdn/nw-sld*.
- You have a user assigned to the security role DataSupplierLD.
- The SLD to which you register Sybase Unwired Platform must be the latest Common Information Model version (currently 1.6.21).

Task

You can find more information on SLD in the document *Configuring, Working with and Administering System Landscape Directory*. Access the SDN link provided above, then search for the latest version of the document.

1. *Configuring the SLD Destination*

   Configure the SLD Destination with Sybase Unwired Platform.
2. *Generating the Payload*

   Generate the payload that contains all details of the installed Sybase Unwired Platform system, and upload the payload to System Landscape Directory (SLD).

Configuring the SLD Destination

Configure the SLD Destination with Sybase Unwired Platform.

Prerequisites

You have installed the Sybase Unwired Platform.

Task

1. Navigate to the installation path `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\SLD\` to locate the files corresponding to the Data Supplier.
2. From the command prompt, run the batch file `runSLDReg.bat`.
3. Enter the following details:

Options	Description
UserName	User name of the SLD system.
Password	Password used to authenticate the SLD system.
ServerHost	IP Address of the SLD Server.
Port	HTTP(S) port on which the SLD server is running.
Use https	Indicate if you want a secure connection or not.
Write this information to secure file	If you want to store the encrypted HTTP connection information, enter `y`.

The configuration (`.cfg`) and key (`.cfg.key`) files are created in the location: `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\SLD\SLDREG`.

Generating the Payload

Generate the payload that contains all details of the installed Sybase Unwired Platform system, and upload the payload to System Landscape Directory (SLD).

Prerequisites

- Ensure all Sybase Unwired Platform services are currently running before you generate the payload.
- You have defined the JAVA_HOME environment variable. For example, if you have installed SUP on your system, set the JAVA_HOME as `set JAVA_HOME=C:\sybase\UnwiredPlatform\JDK1.6.0_16`

Task

1. From the command prompt, run the batch file `runXMLgenerator.bat`.
2. Enter the following details:

Options	Description
Login name	Sybase Unwired Platform administrator's login name
Login password	Sybase Unwired Platform administrator's login password

If the payload generation is successful, the payload file `SUP_PayLoad.xml` is created in the location `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\SLD\SLDREG\`.

3. To upload the payload to SLD, run the batch file `sendPayLoad.bat`.

CHAPTER 4 Device and Application Provisioning Overview

Provisioning describes all activities used to setup the device and its applications, so it can interact with the Unwired Platform environment.

Device and application provisioning includes:

1. Distributing and installing the platform infrastructure components. Devices that require provisioning of infrastructure components include:
   - Afaria clients for Windows, Windows Mobile, and BlackBerry devices.
   - Client runtime for messaging-based synchronization applications for Windows and Windows Mobile. No runtime is required for iOS.
   - Security artifacts for encryption and SSO. See *Device Security* in the *Security* guide.
2. Distributing and installing the device application that interacts with packages that are deployed to the Unwired Server. For heterogeneous device provisioning support, Sybase recommends Afaria.
3. If you are using a replication application, setting up the device for push synchronization. See *Setting up Push Synchronization for Replication*.
4. If you are using a messaging application, registering the user devices using Sybase Control Center (each user then activates using the Sybase Settings application deployed in step 1).

See also

- *Setting up Push Synchronization for Replication* on page 63

Provisioning Methods by Application Type

Depending on the application type you are deploying, there are different methods to ensure all application and runtime artifacts are downloaded over-the-air (OTA), then stored or installed on the device, without requiring a physical connection the corporate LAN.

Review the various options you can use. Options are organized according to the application type and device target. For larger client bases, Sybase recommends an OTA method where the artifacts are pushed to the device, or are pulled by the user via a URL or link. Generally speaking, hard connection options are best reserved to small deployments or development testing purposes.

Device Application	Method	Details
BlackBerry native – once applications provisioned and installed, 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.	BlackBerry Enterprise Server (BES) push notifications	When the BlackBerry device activates, it automatically pairs with the BES and downloads the application. See *http://www.blackberry.com/btsc/search.do?cmd=displayKC&docType=kc&externalId=KB03748* for step-by-step instructions. Also see *BlackBerry Provisioning with BES* in *System Administration*.
	URL/link to installation files	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 JAD file.
iOS native – users of iPhones tend to self-manage their devices: device users download application files to their iOS devices and synchronize updates as required. As Administrator, you must ensure users are notified of required downloads.	Apple Push Notification Service (APNS)	APNS allows users to receive notifications on iPhones. Each application that supports APNS must be listed in Sybase Control Center with its certificate and application name. See *Provisioning with iOS APNS* in *System Administration*.
Messaging native applications – may require direct connections either on the LAN or WIFI. This connection prevents man-in-the-middle attacks that might otherwise occur.	Direct Unwired Server connections	If you are not using Afaria for messaging clients, you must first install the client application then connect to corporate LAN using WIFI or any other method of your choosing in order to provision devices with required files. See *Provisioning with Unwired Server* in *System Administration*.

Device Application	Method	Details
Hybrid Web Container – because this native application embeds a browser control supplied by the device OS, these devices require specific provisioning techniques.	Configuration Files	The Hybrid Web Container supports only the provisioning from a file method. Provisioning of the Hybrid Web Container provisions the container itself (and therefore all the apps in the container). See *Provisioning with Configuration Files* in *System Administration*.
Heterogeneous applications (including online data proxy and DOEC) –if you are supporting a heterogeneous device environment, these methods are the best way to provision all devices using the same method. Which method you choose depends on whether or not you want devices connecting directly to Unwired Server.	Afaria and Configuration Files	Automate the process of provisioning applications by enabling a group of applications to be initially provisioned with a set of parameters from a configuration file using Afaria. See *Provisioning with Afaria* in *System Administration*.
Windows, Windows Mobile, and BlackBerry	Desktop Management and cradle	Installs the native application when the device is synchronized via a computer. See your device documentation for details.

See also

- *Provisioning with a Configuration File* on page 39
- *Provisioning with Afaria* on page 45
- *iOS Provisioning with APNS* on page 54
- *BlackBerry Provisioning with BES* on page 58
- *Provisioning with Unwired Server* on page 62
- *Setting up Push Synchronization for Replication* on page 63

Provisioning with a Configuration File

In environments where Afaria is not used for mobile device management, you can use a file to provision the device application on BlackBerry, Android devices, or Windows Mobile or

Win32 devices. iOS clients cannot be manually provisioned in this manner, as the application sandbox is not accessible.

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

1. *Provisioning the Unwired Server Public Key*

   (Optional) Retrieve the public key of the Unwired Server to populate its value in the provisioning file.
2. *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.
3. *Deploying the Provisioning File*

   Make the provisioning file available to the device so that it can be automatically deployed when the application starts.
4. *Next Steps*

   After you have provisioned the application, perform automatic or manual registration of application connections to associate users to the application.

See also

- *Provisioning Methods by Application Type* on page 37
- *Provisioning with Afaria* on page 45
- *iOS Provisioning with APNS* on page 54
- *BlackBerry Provisioning with BES* on page 58
- *Provisioning with Unwired Server* on page 62
- *Setting up Push Synchronization for Replication* on page 63

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 `ServerVerificationKey.txt` in `<UnwiredPlatform_InstallDir>/Servers/MessagingServer`. It is created only at messaging service startup, so the messaging service must be started at least once to see this file. The data in this file is the Base64 encoded server verification key and should 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.

1. Create a text file, named as follows:
   - For provisioning Android device applications, the file must be named "Sybase_Messaging_" followed by the package name with a ".cfg" extension. Any non-alphanumeric characters in the application ID will be replaced with underscores to avoid any conflicts with operating system restrings. For example, if the package name is "myApplication:1.0", the file would be: `Sybase_Messaging_myApplication_1_0.cfg`.
   - For provisioning BlackBerry device applications, the file must be named "Sybase_Messaging_" followed by the Application ID with a ".cfg" extension. Any non-alphanumeric characters in the Application ID will be replaced with underscores to avoid any conflicts with operating system restrings. For example, if the Application ID is "myApplication:1.0", the file would be: `Sybase_Messaging_myApplication_1_0.cfg`.
   - For provisioning Windows Mobile or Win32 device applications, the file must be named "Sybase_Messaging_" followed by the Application ID with a ".cfg" extension. Any non-alphanumeric characters in the Application ID will be replaced with underscores to avoid any conflicts with operating system restrings. For example, if the Application ID is "myApplication:1.0", the file would be: `Sybase_Messaging_myApplication_1_0.cfg`.
   - For provisioning the Hybrid Web container used for Mobile Workflow, the file must be named "Sybase_Messaging_", followed by the application ID (HWC), with a `.cfg` extension. Any non-alphanumeric characters in the application ID are replaced with underscores to avoid any conflicts with operating system restrings. For example, if the Mobile Workflow package application ID is "myApplication:1.0," the file would be: `Sybase_Messaging_myApplication_1_0.cfg`.

   Note: When configuring the connection settings for the Hybrid Web Container, settings are applied for the Hybrid Web Container. These settings are not per mobile workflow application.
2. Enter the desired settings information in the `.cfg` file. See *Provisioning Settings*.
3. Save the .cfg file.

See also

- *Deploying the Provisioning File* on page 43

Provisioning Settings

Enter settings in the provisioning file for connecting to Unwired Server.

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 used, where the mobile application project is deployed.
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's public key. An invalid Base64 string results in an empty key.

Key	Description
autoreghint=	This value is used exclusively for the SUP administrator to convey information to the device application; Unwired Server runtime does not use this value. The application developer uses this setting at runtime to indicate how the application is initially registered – either automatically or manually. • 1 – Always use automatic application registration. The application user is not prompted to enter an activation code upon initial activation. • 2 – Always use manual device registration. The application user is prompted to enter an activation code upon initial activation. This implies that automatic device application registration is not enabled on Unwired Server. • 0 – Both modes are supported. The application user interface allows automatic registration , or requires an activation code for manual registration.
urlsuffix=	The URL suffix is a pattern that is used internally by Sybase Unwired Platform when constructing URLs as part of the correct path to connect to Unwired Server. By default, this property does not need to be set. The device client internally 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:** You only need to explicitly configure this property to match the modified URL suffix, if the default path and URL suffix of the Relay Server is modified.

Deploying the Provisioning File

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

1. Copy the provisioning file to the device:

- Android devices: 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.").
- BlackBerry devices: 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 – client supports provisioning only through Afaria client deployment.
- Windows Mobile devices: 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 devices: 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).
- When provisioning the Hybrid Web Container on a device:
  - Android – the Android client searches the root of the flash card ("/sdcard/"), followed by the client file root ("/data/data/com.sybase.workflow/files.").
  - BlackBerry – the BlackBerry client searches the root of the flash card ("fileconn.dir.memorycard"), followed by the client file root ("file:///store/home/user/").
  - iOS – client supports provisioning only through Afaria client deployment.
  - Windows Mobile – the Windows Mobile client searches the root of any flash cards, followed by the root of the file system.

2. Register the application connection:
   - For device clients, log in to Sybase Control Center and register the application connection as described in the topic *Registering and Reregistering a Connection* in *System Administration*.
   - For Mobile Workflow clients, log in to Sybase Control Center and register the Mobile Workflow application connection as described in the topic *Registering and Reregistering Mobile Workflow Application Connections* in *Developer Guide: Mobile Workflow Packages*.
3. Start the application.
   When the application is started, a search is performed for the provisioning file only if the device is not yet configured. Unconfigured is defined as missing one or more of the required connection properties (server name, port, user name, activation code, and so on). If unconfigured, the settings from the provisioning file are automatically applied.

See also

- *Creating a Provisioning File* on page 41

Next Steps

After you have provisioned the application, perform automatic or manual registration of application connections to associate users to the application.

For information on the registration process, and registration procedures, see *Sybase Control Center for Sybase Unwired Platform > Setting Up Application and User Connections*.

See also

- *Provisioning Configuration Data and Certificates* on page 53

Provisioning with Afaria

(Applies only to Online Data Proxy) 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 using Afaria.

1. *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.
2. *Prerequisites for Provisioning an Application Using Afaria*

   (Applies only to Online Data Proxy) Verify that these prerequisites for provisioning an application using Afaria are completed by the IT admin and application developer.
3. *Preparing the Provisioning File*

   (Applies only to Online Data Proxy) Create the provisioning file required for automatic provisioning of an application, and provide it to the Afaria administrator.
4. *Provisioning the Unwired Server Public Key*

   (Applies only to Online Data Proxy) Retrieve the public key of the Unwired Server to populate its value in the provisioning file.
5. *Creating a Provisioning File*

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

   (Applies only to Online Data Proxy) 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.
7. *Next Steps*

After you have provisioned the application, perform automatic or manual registration of application connections to associate users to the application.

See also

- *Provisioning Methods by Application Type* on page 37
- *Provisioning with a Configuration File* on page 39
- *iOS Provisioning with APNS* on page 54
- *BlackBerry Provisioning with BES* on page 58
- *Provisioning with Unwired Server* on page 62
- *Setting up Push Synchronization for Replication* on page 63

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.

1. *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, rather than needing cradle the device and connect to the network. OTA deployment is recommended for production environments that require you to administer many device users.
2. *Configuring Afaria Server*

   Configure Afaria Server to use the same gateway as the OTA Deployment Center. You can optionally configure email as well.
3. *Creating Addresses, Groups, and Profiles*

   Session Manager allows you to control each user session opened with the SMS or email download notification and determine what actions are subsequently performed.
4. *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.

5. *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 step makes Afaria a managed resource of Sybase Control Center, and allows you launch and use Afaria remotely from this Web console.

See also

- *Prerequisites for Provisioning an Application Using Afaria* on page 49

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, rather than needing cradle the device and connect to the network. OTA deployment is recommended for production environments that require you to administer many device users.

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 email 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 email 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 determine what actions 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 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. Use program called OTA 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 step makes Afaria a managed resource of Sybase Control Center, and allows you launch and use Afaria remotely from this Web console.

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, you choose to install Sybase Control Center and Afaria Server on the same host you can use the following connection properties:

Options	Description
Host	localhost
Port	80

   Otherwise, use the values for the host of the Afaria Server. In most production environments where Afaria is already installed, this will likely be the case. The Afaria server is added to the **Perspective Resources** window, 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

(Applies only to Online Data Proxy) Verify that these prerequisites for provisioning an application using Afaria are completed by the IT admin and application developer.

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. See topics in the *Developer Guide: OData SDK*.

- *Provisioning Connection Settings from Afaria*
- *Provisioning Certificates using Afaria*
- *Prompting User for Required Configuration Information*

2. Verify that the application developer provides the application ID or package name of the application to the Sybase Unwired Platform administrator for use in naming the provisioning file.
3. Verify that the IT 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.

See also

- *Setting Up the Afaria Environment* on page 46

Preparing the Provisioning File

(Applies only to Online Data Proxy) 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 Unwired Server. See *Creating a Provisioning File*.
2. Provides the file(s) to the Afaria Administrator for import into the Afaria portal package.
3. Register the application connection:
   - For device clients, log in to Sybase Control Center and register the application connection as described in the topic *Registering and Reregistering a Connection* in *System Administration*.
   - For Mobile Workflow clients, log in to Sybase Control Center and register the Mobile Workflow application connection as described in the topic *Registering and Reregistering Mobile Workflow Application Connections* in *Developer Guide: Mobile Workflow Packages*.

   When the application connects to Unwired Server, you can view the application connection in the Application Connections tab in Sybase Control Center.

Provisioning the Unwired Server Public Key

(Applies only to Online Data Proxy) 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 `ServerVerificationKey.txt` in `<UnwiredPlatform_InstallDir>/Servers/MessagingServer`. It is created only at messaging service startup, so the messaging service must be started at least once to see this file. The data in this file is the Base64

encoded server verification key and should be used as the value for the `serververificationkey=` key.

To ensure proper security for application clients on the iOS, Android and Blackberry platforms, pre-provision the Unwired Server public key:

1. Perform onboarding of devices securely from within the 'intranet' in the corporate firewall by directly providing the connection details for the relay server (RS) which is present in the DMZ.

   Network traffic from the device to DMZ Relay server is not routed through the public internet because typically there is an outbound port opened between the corporate network and the DMZ network to enable a secure communication. There is no forward proxy setting involved.

   Sybase Unwired Platform persists the public key obtained during onboarding 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 onboard with a different Unwired Server, the application explicitly must use the clear or reset APIs to clear the verification key, as described in *Developer Guide: OData SDK*. Alternatively, end users can uninstall and re-install the application.

Another option is for applications to pre-provision the Unwired Server public key securely onto the device before onboarding from a public network. See *System Administration > Device Provisioning > Application Provisioning > Provisioning an Application Using a File*.

Creating a Provisioning File

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, named as follows:
   - For provisioning iOS device applications, the file must be named "Sybase_Messaging_" followed by the Application ID with a ".cfg" extension. Any non-alphanumeric characters in the Application ID will be replaced with underscores to avoid any conflicts with operating system restrings. For example, if the Application ID is "myApplication:1.0", the file would be: `Sybase_Messaging_myApplication_1_0.cfg`.
   - For provisioning Android device applications, the file must be named "Sybase_Messaging_" followed by the package name with a ".cfg" extension. Any non-alphanumeric characters in the application ID will be replaced with underscores to avoid any conflicts with operating system restrings. For example, if the package name is "myApplication:1.0", the file would be: `Sybase_Messaging_myApplication_1_0.cfg`.

Note: When configuring the connection settings for the Hybrid Web Container, settings are applied for the Hybrid Web Container. These settings are not per mobile workflow application.

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

See also

- *Provisioning Configuration Data and Certificates* on page 53

Provisioning Settings

Enter settings in the provisioning file for connecting to Unwired Server.

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 used, where the mobile application project is deployed.
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's public key. An invalid Base64 string results in an empty key.

Key	Description
autoreghint=	This value is used exclusively for the SUP administrator to convey information to the device application; Unwired Server runtime does not use this value. The application developer uses this setting at runtime to indicate how the application is initially registered – either automatically or manually. • 1 – Always use automatic application registration. The application user is not prompted to enter an activation code upon initial activation. • 2 – Always use manual device registration. The application user is prompted to enter an activation code upon initial activation. This implies that automatic device application registration is not enabled on Unwired Server. • 0 – Both modes are supported. The application user interface allows automatic registration , or requires an activation code for manual registration.
urlsuffix=	The URL suffix is a pattern that is used internally by Sybase Unwired Platform when constructing URLs as part of the correct path to connect to Unwired Server. By default, this property does not need to be set. The device client internally 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:** You only need to explicitly configure this property to match the modified URL suffix, if the default path and URL suffix of the Relay Server is modified.

Provisioning Configuration Data and Certificates

(Applies only to Online Data Proxy) 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 iOS Application* *Provisioning Configuration Data for an Android 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 iOS Application* *Provisioning a Certificate for an Android 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.

See also

- *Creating a Provisioning File* on page 51

Next Steps

After you have provisioned the application, perform automatic or manual registration of application connections to associate users to the application.

For information on the registration process, and registration procedures, see *Sybase Control Center for Sybase Unwired Platform > Setting Up Application and User Connections*.

See also

- *Provisioning Configuration Data and Certificates* on page 53

iOS Provisioning with APNS

Unlike the Afaria method of device provisioning, Apple provisioning is primarily performed by developers at the end of their development cycle, with some Sybase Control Center

configuration performed by the administrator, and some device setup by the device user. Apple Push Notification Service (APNS) supports user notifications on iPhone devices.

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 on iPhones. APNS:

- Works only with iPhone physical devices
- Is not required for any iOS application
- Cannot be used on an iPhone simulator
- Cannot be used with iPod touch or iPad devices
- Must be set up and configured by an administrator on the server
- Must be enabled by the user on the device

See also

- *Provisioning Methods by Application Type* on page 37
- *Provisioning with a Configuration File* on page 39
- *Provisioning with Afaria* on page 45
- *BlackBerry Provisioning with BES* on page 58
- *Provisioning with Unwired Server* on page 62
- *Setting up Push Synchronization for Replication* on page 63

Configuring Apple Push Notification Service

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

Prerequisites

The following prerequisites must be performed by the developer before the administrator can configure the Apple Push Notification Service (APNS):

- 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's 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 they 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 key chain 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 Apple Push Notification Service (APNS) push to work, because APNS push functionality stops working on 22 December 2010.

Apple now 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.*
- Used APNS initialization code in the codeline.

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. Copy the enterprise certificate (`*.p12`) to the computer on which Sybase Control Center has been installed. Save the certificate in *`UnwiredPlatform_InstallDir`*`\Servers\MessagingServer\bin\`.
3. In Sybase Control Center, expand the **Servers** folder and click **Server Configuration** for the primary server in the cluster.
4. In the **Messaging** tab, select **Apple Push Configuration**, and:

a) Configure Application name with the same name used to configure the product name in Xcode. If the certificate does not automatically appear, browse to the directory.
b) Change the push gateway information to match that used in the production environment.
c) Restart Unwired Server.

5. Verify that the server environment is set up correctly:
   a) Open *UnwiredPlatform_InstallDir*`\Servers\UnwiredServer\logs\APNSProvider`.
   b) Open the log file that should now appear in this directory. The log file indicates whether the connection to the push gateway is successful or not.
6. Deploy the application and the enterprise distribution provisioning profile to your users' computers.
7. Verify that the APNS-enabled iOS device is set up correctly:
   a) Click **Device Users**.
   b) Review the Device ID column. The application name should appear correctly at the end of the hexadecimal string.
   c) Select the Device ID and click **Properties**.
   d) Check that the APNS device token has been passed correctly from the application by verifying that a value is in the row. A device token appears only after the user is registered with the application in Sybase Control Center.
8. 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. Allow a few minutes for the delivery or notification mechanism to take effect and monitor the pending items in the Device Users data to make sure the value increases appropriately for the applications.
9. To troubleshoot APNS, use the *UnwiredPlatform_InstallDir*`\Servers\Unwired Server\log\trace\APNSProvider` log file. You can increase the trace output by editing `<SUP_Home>\Servers\MessagingServer\Data\TraceConfig.xml` and configuring the tracing level for the APNSProvider module to debug for short periods.

See also

- *Configuring Push Notifications for the BlackBerry Enterprise Server* on page 59
- *Configuring Messaging Synchronization Properties* on page 17

BlackBerry Provisioning with BES

BlackBerry devices can be provisioned with BlackBerry servers in addition to the Afaria method.

BlackBerry devices that are connected to a production environment using relay server can use BlackBerry Enterprise Server (BES) to provision supported device types.

See also

- *Provisioning Methods by Application Type* on page 37
- *Provisioning with a Configuration File* on page 39
- *Provisioning with Afaria* on page 45
- *iOS Provisioning with APNS* on page 54
- *Provisioning with Unwired Server* on page 62
- *Setting up Push Synchronization for Replication* on page 63

Provisioning Prerequisites for BlackBerry

Complete the prerequisites before provisioning the application.

First, look at the prerequisites for all device platforms, then this information for BlackBerry-specific configuration details.

Sybase Unwired Platform Installation and Configuration

Prerequisite	Where to Find Information
Install a Sybase Relay Server if Unwired Server is behind the internal firewall. **Note:** The use of a Relay Server is optional; however, if you want to use one in your Unwired Platform system configuration, you would install and configure it prior to provisioning the application to the device.	*Installation Guide for Runtime*

BlackBerry Enterprise Server Configuration

Prerequisite	Where to Find Information
Configure push notifications from Unwired Server to each BES.	*Configuring BlackBerry Push Settings*

Prerequisite	Where to Find Information
Add 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*

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.	*Runtimes and Clients* as well as the Device User Guide for any SAP solution you are deploying.
Provide enterprise activation information so device users can pair their devices to the BlackBerry Enterprise Server.	*http://na.blackberry.com/eng/support/enterpriseactivation/*
Install BlackBerry Desktop Software on a personal computer, which includes Desktop Manager, if your organization is installing the application on a small number of devices. **Note:** Desktop Manager is required only if your device users will use it to install the application.	*http://na.blackberry.com/eng/services/desktop/*

See also

- *Configuring Push Notifications for the BlackBerry Enterprise Server* on page 59

Configuring Push Notifications for the BlackBerry Enterprise Server

Configure push notifications from Unwired Server to the BES using Sybase Control Center.

BlackBerry push notifications alert offline users to the availability of new items awaiting retrieval on Unwired Server. Push 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. You can also enable the Push Access Protocol (PAP) if you are using a Push Proxy Gateway to deliver messages.

You configure the notifications for each BES.

1. Log into Sybase Control Center.

2. Expand **Servers** and select the appropriate server.
3. Select **Server Configuration**.
4. In the right administration pane, select the Messaging tab, then select the **BlackBerry Push Notification** tab.
5. Select **New**, then enter all applicable information.

   The URL takes the format:

   ```
   http://<DNS or IP address>:<port number>
   ```

   The default port number is 8080.

See also

- *Configuring Apple Push Notification Service* on page 55
- *Configuring Messaging Synchronization Properties* on page 17

Provisioning 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	Purpose	Description
BlackBerry Enterprise Server (BES) Over-the-Air (OTA)	Enterprise installations	When the BlackBerry device activates, it automatically pairs with the BES and downloads 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 `SybaseMobileWorkflow.zip` file needed for deployment is located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\ClientAPI\Workflow\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/*. The `Workflow.ALX` and supporting COD files for deployment are located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\ClientAPI\Workflow\BB`.
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 JAD file. The `JAD` and associated files for this type of deployment are located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\ClientAPI\Workflow\BB\OTA`.
Desktop Manager	Personal installation	Installs the application when the BlackBerry device is synced via a computer.

Provisioning Method	Purpose	Description
BlackBerry App World	Evaluation/Demo	Downloads a solution-specific application in demo mode. Demo mode runs standalone, and the device does not connect to Sybase Unwired Platform. *http://appworld.blackberry.com/webstore/*

Provisioning with Unwired Server

If you are not using Afaria, you can install the client application then connect to corporate LAN using WIFI or any other method of your choosing in order to provision devices with required files.

Sybase recommends you follow this method when you are not using Afaria. This method ensures that the public RSA key required for future secure communication is correctly and reliably installed.

1. Install the device client application to the device.
2. Connect to the corporate LAN upon which Unwired Platform 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 it is accessible from the corporate LAN.
4. The messaging service on Unwired Server seeds the client with the public key.

   The client uses this public key for all subsequent connections.
5. Provide the user with instructions to re-configure the connection properties on the device to use Relay Server from the Internet for subsequent connections.

See also

- *Provisioning Methods by Application Type* on page 37
- *Provisioning with a Configuration File* on page 39
- *Provisioning with Afaria* on page 45
- *iOS Provisioning with APNS* on page 54
- *BlackBerry Provisioning with BES* on page 58
- *Setting up Push Synchronization for Replication* on page 63

Setting up Push Synchronization for Replication

(Not applicable to Online Data Proxy)If your device application requires push synchronization, you must configure the device for server-initiated synchronization.

Prerequisites

The device must already be provisioned with Unwired Platform runtimes. You also must have also enabled push synchronization for the Unwired Server as well as configured cache refresh triggers for the mobile business object in Sybase Control Center. For details, see Sybase Control Center online help:

- *Enabling Push Synchronization*
- *Scheduling a Cache Refresh*

Task

1. The developer creates the application and designs it as a push synchronization application.
2. The user starts the application on the device or emulator.
3. The user or administrator enables push sync on the device or emulator and set the push notification role.
4. The user synchronizes the MBO to Unwired Server, and restarts the listener so future synchronizations push to the device correctly.

See also

- *Provisioning Methods by Application Type* on page 37
- *Provisioning with a Configuration File* on page 39
- *Provisioning with Afaria* on page 45
- *iOS Provisioning with APNS* on page 54
- *BlackBerry Provisioning with BES* on page 58
- *Provisioning with Unwired Server* on page 62

CHAPTER 5 Application and User Management Overview

The goal of application management is to register an application to Unwired Server as an entity, create an application template that specifies application connection details for a user, and activate applications either manually or automatically.

Table 4. Application and user management tasks

Task	Frequency	Accomplish by using
Create new applications to register application entities with Unwired Server. A default application template is created automatically. Modify and delete applications as part of application life cycle.	As required	Sybase Control Center for Unwired Platform with Applications node, and Applications tab.
Create or modify application connection templates to specify details for native, workflow, and proxy application connections.	As required	Sybase Control Center for Unwired Platform, with Application node, and Application Connection Templates tab.
For applications that need to be registered manually, register an application connection to associate an application connection with a user. This is not necessary for applications that are registered automatically.	As required	Sybase Control Center for Unwired Platform, with Application node, and Application Connections tab.
View activated users, once they have logged in with the activation code. Users must either supply the activation code manually, or the device client supplies the activation code automatically as coded.	As required	Sybase Control Center for Unwired Platform with the Application node, and Application Users tab.
Create a new activation code for a user whose code has expired.	As required	Sybase Control Center for Unwired Platform, with Application node, and Application Connections tab.
Review registered application connections and users, delete application connections to free licenses, delete application connections to remove users from the system	As required	Sybase Control Center for Unwired Platform with the Applications node.

Task	Frequency	Accomplish by using
Manage subscriptions	As required	Sybase Control Center for Unwired Platform with the Packages node.

Information and guidelines:

Note: Application connection is not used for native replication applications at this time.

- Two activation options are available for onboarding, which refers to the process of activating an authentic device client, user, and application entity, as a combination, in Unwired Server:
  - Automatic activation – requires a user to present credentials to use the application on a supported device client.
  - Manual activation – requires the user to present the activation code upon log on to a supported device client. The system administrator establishes an activation code when registering the application connection for the user.
- Native messaging applications can only be activated manually.
- Application template are used for automatic activation. Therefore, when setting up the application template for automatic registration, be sure to set up the security configuration, domain, the application ID, and automatic registration enabled properties in application settings. Those are used for automatic application registration.

  When a client application connects to the server with its application ID and credentials, and requests automatic registration, the application ID is used to look up a matching template. If that template allows automatic registration (the Automatic Registration Enabled property is set to true),the security configuration in the template is used to validate the credentials. Upon successful validation of those credentials, the user identity is registered in the Unwired Server. The client application may also include the security configuration as part of the username (user@securityconfiguration) and in that case, the security configuration (in addition to application ID) is used to look up a matching template. If no or multiple templates are detected, the registration request fails.
- Supported device client activation options:

Device Client Type	Automatic Registration	Manual Registration
Workflow messaging	X	X
Native messaging		X
Online Data Proxy	X	X

Applications

An Application is the runtime entity that can be directly correlated to a native or mobile workflow application. The application definition on the server establishes the relationship

among packages used in the application, domain that the application is deployed to, user activation method for the application, and other application specific settings.

- For native replication/messaging applications, one or more MBO packages can be assigned to an application. If the application developer uses the same package in a different application, the MBO package must be assigned to that application.
- For the mobile workflow applications, all MBO packages are accessible in all domains, so no MBO packages need to be assigned to the mobile workflow application (HWC).
- For Online Data Proxy applications, no MBO package assignments are needed as well.

Applications are managed and monitored by administrators on the Sybase Control Center. They are created automatically or manually through Sybase Control Center.

An application ID uniquely identifies an application to Unwired Server. Application connection templates enable administrators to manually register application connections in Unwired Server with predefined settings. Templates also enable automatic activation of devices (described later). Users are associated with one or more applications through application connections. Administrators can view application users in Sybase Control Center as soon as a user logs onto the application from a device.

Application Users

In Unwired Platform, an application user is an identity registered as the user of one or more versions of a device application. Application users are managed in Sybase Control Center.

For native replication applications, an application user is automatically registered upon first successful authentication using the security configuration associated with the package that user is trying to access. For native messaging applications, an application user is automatically registered upon first successful synchronization after manual registration. For workflow and Online Data proxy applications, the application user is registered upon successful registration whether manual or automatic. A user can have multiple devices.

The platform administrator can view the user name and the security configuration that was used to authenticate the user. In addition, the administrator can remove users that no longer exist.

Note: SAP DOE-C package users are not registered in Unwired Server. Those users are authenticated by their respective DOE back-end servers.

Application Creation

There are two ways an application gets created - automatic and manual.

Automatic Application Creation

Applications are created automatically, when the system administrator deploys a package. The mobile workflow application (HWC) is created during Unwired Server installation.

An application is created automatically when an MBO package is deployed to the server. In case of upgrade from a previous server version, an application is created for all deployed MBO packages. The default name of an application is the same as the package name.

Note: An application is primarily used for tracking purpose. At this point, an application connection template is also created, but not used at this time since automatic registration is not available for native applications (MBO package client applications).

For Online Data Proxy, there is no such automatic creation of applications. Applications must be created manually.

Manually Creating Applications

Create an application manually by assigning a unique application ID and other key application properties, such as domain, MBO package, security configuration, among others. At this time, the manual process is only needed for Online Data Proxy applications or when using a Hybrid Web Container built using the iOS sample, where developers can use their own application IDs for workflow applications.

Launching the Application Creation Wizard

Use the Application Creation wizard to register an application.

1. In the left navigation pane of Sybase Control Center, click the **Applications** node and select the **Applications** tab in the right administration pane.
2. To register an application, click **New**.
   The Application Creation wizard is displayed.

Setting General Application Properties

Provide general application properties such as the application ID, description, security configuration and domain details while registering the application.

1. In the Application Creation Wizard, enter a unique **Application ID**, following application ID guidelines.
2. Enter a **Display name** and **Description** for the application.
3. Select the appropriate security configuration from the **Security Configuration** drop-down list.
4. Select the appropriate domain from the **Domain** drop-down list.
5. (Optional) Select a package from the list to assign an application connection template.
   a) Select **Configure additional settings**, and click **Next**.
   b) In the Application Creation wizard, enter a **Template name** and **Description** for the application connection template.
   c) To reuse the configuration of an existing template, select a **Base template** from the drop-down list.

d) Configure the following property categories as required.
   - Apple Push Notifications
   - Application Settings
   - BlackBerry Push Notifications
   - Connection
   - Custom Settings
   - Device Advanced
   - Device Info
   - Proxy
   - Security Settings
   - User Registration

6. Click **Finish** to register the application with the configured settings.

See also

- *Application Connection Properties* on page 317
- *Application ID Overview* on page 70
- *Application ID Guidelines* on page 71

Application Connections

Application connections associate an application instance with a user. An application may be used by many users, and a user may be associated with many applications.

See the *Applications Connections* topic group in *Sybase Control Center* online help for additional information and procedures.

Registering or Reregistering Application Connections

Use Sybase Control Center to trigger the registration of an application connection, or reregister an application connection when the activation code has expired.

1. In the left navigation pane, click the **Applications** node.
2. In the right administration pane, click the **Application Connections** tab.
3. Choose an action:
   - Click **Register** to register a new application connection. Using the Activation Code, this application is then paired with a user and a device.
   - Click **Reregister** to associate the application with a new device and user pairing. For example, reregister the application connection if someone loses their device. By reregistering the application connection, the user then receives the same applications and workflows as the previous device.
4. In the Register Application Connection or the Reregister Application Connection dialog.

a) For new device registration only, type the name of the user that will activate and register the device. For reregistrations or clones, the same name is used and cannot be changed.
b) Select the name of the template for initial application connection registration. 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 mobile workflows (containers). Use as is, or customize. If you use the HWC template, Application ID must be set to HWC.
   - Custom - customized templates are listed.

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.

   If you are using Relay Server, ensure the correct values are used.
   - **Server name**- the DNS name or IP address of the primary Unwired Server, such as "myserver.mycompany.com". If using Relay Server, the server name is the IP address or fully qualified name of the Relay Server host.
   - **Port**- the port used for messaging connections between the device and Unwired Server. If using relay server, this is the Relay Server port. Default: 5001.
   - **Farm ID**- a string associated with the relay server farm ID. Can contain only letter A-Z (uppercase or lowercase), numbers 0-9, or a combination of both. Default: 0.
   - **Application ID**- the application ID registered for the application. The value differs according to application client type - native messaging, workflow, or Online Data Proxy client. See *Application ID Overview* for guidelines.
   - **Security Configuration**- select the security configuration relevant for the application connection.
   - **Domain**- select the domain for which you want to register the application connection with.
   - **Activation code length** - the number of characters in the activation code. If you are reregistering or cloning a device, this value cannot be changed.
   - **Activation expiration**- the number of hours the activation code is valid.
6. (Optional) Select the check box adjacent to **Specify activation code** to enter the code sent to the user in the activation e-mail. This value can contain letter A - Z (uppercase or lowercase), numbers 0 - 9, or a combination of both. Acceptable range: 1 to 10 characters.
7. Click **OK**

Application ID Overview

Applications can be directly correlated to a native application or mobile workflow container instance on device. A native application is the single binary deployed to device which may use one or more MBO packages. The mobile workflow application is a collection of workflow packages and constitutes as one application. One or more MBO packages can be assigned to an application. If application developers want to use the same package in a different application, they can do that by assigning the MBO package to that application using Sybase Control Center.

An application ID uniquely identifies the application and must be used to register an application connection and also used in the device application in case of Online Data Proxy client for activation of its application connection.

See also

- *Application ID Guidelines* on page 71
- *Application Connection Properties* on page 317
- *Setting General Application Properties* on page 68

Application ID Guidelines

Follow these guidelines for choosing an appropriate application ID while registering application connection for use by native messaging, workflow client, or Online Data Proxy. Failure to specify the correct application ID would result in failure when the client tries to activate itself even though the user name and activation code do match.

Registration Type	Application ID Guidelines
Native messaging client	The application ID should always be left empty. If using a preexisting registration template (such as Default), then the application id would be empty already.
Native Android client application	An application ID is required. Register the application connection using the template created for the application.
Workflow client	• 2.0.1 or earlier version – leave the application ID empty. • 2.1 or later version – use preexisting "HWC" template, or, if using your own template, make sure that "HWC" is set as the application ID in the template. • iOS Sample container 2.1 version – use the template you have created. The application ID used by the iOS sample container should match the application ID specified in registration.
Online Data Proxy client	Register application connection using the template created for the application. Existing templates can be found in the **Applications > Application Connection Template** tab.

See also

- *Application ID Overview* on page 70
- *Application Connection Properties* on page 317
- *Setting General Application Properties* on page 68

Application Connection Templates

An application connection template is a model or pattern used to standardize connection properties and values for a specific application connections so that they can be reused. A template allows you to quickly create actual application connections.

Two default templates are available: Default and HWC. The Default is for registering an application connection without an application ID (backward compatibility scenario, as well as for 2.1.1 native messaging clients). The HWC template is for registering application connections for 2.1 or later version of Hybrid Web Container clients.

See the *Applications Connection Templates* topic group in *Sybase Control Center* online help for additional information and procedures.

Creating Application Connection Templates

Create application connection templates by setting approiate properties and values.

1. In the left navigation pane, click the **Applications** node.
2. In the right administration pane, click the **Application Connection Templates** tab.
3. Click **New**.
4. Enter the **Template name** and **Description** for the application connection template.
5. Select the **Base template** from the drop-down list.
6. You can configure any of the following profiles. See *Appliation Settings*:
   - Apple Push Notifications
   - Application Settings
   - BlackBerry Push Notifications
   - Connection
   - Custom Settings
   - Device Advanced
   - Device Info
   - Proxy
   - Security Settings
   - User Registration
7. Click **OK**.

See also

- *Application Connection Properties* on page 317

CHAPTER 6 MBO Package Management Overview

The goal of mobile business object (MBO) package management is to make MBOs available to device users. MBO package management typically requires a one-time deployment and configuration, except for ongoing subscription management for messaging and Data Orchestration Engine connector (DOE-C) packages.

A package, along with its current settings for cache groups, role mappings, synchronization groups, connections, and security configuration, can be exported to an archive and imported back into Sybase Control Center for backup or to facilitate a transition from a test environment to a production environment.

Table 5. MBO package management tasks

Task	Package type	Frequency	Accomplish by using
Deploy packages to a development or production Unwired Server	UNIFIED	Once, unless a new version becomes available	Sybase Control Center for Unwired Platform with the domain-level Packages node
Control user access by assigning security configurations for each package, and mapping roles if fine-grained authorization is enforced through logical roles	UNIFIED	Once, unless security requirements of the package change	Sybase Control Center for Unwired Platform with the domain-level Packages node
Set up the package cache interval and cache refresh schedule (for getting data updated on the Unwired Server from the data source)	UNIFIED	Once, unless data refreshes need to be tuned	Sybase Control Center for Unwired Platform with the domain-level Packages node
Manage subscriptions (UNIFIED, and DOE-C), synchronization groups (UNIFIED), and device notifications (UNIFIED) to customize how updated data in the cache is delivered to the device user	Varies	Periodic, as required	Sybase Control Center for Unwired Platform with the domain-level Packages node

Task	Package type	Frequency	Accomplish by using
Export or import an MBO package	UNIFIED	On-demand, as required	Sybase Control Center for Unwired Platform with the domain-level Packages node
Review current/historical/performance metrics	All	Routine	Sybase Control Center for Unwired Platform with the Monitor node (available only to administrators)
View asynchronous operation replays for the selected package	Replication, UNIFIED	Periodic, as required	Sybase Control Center for Unwired Platform with the domain-level Packages node. However, asynchronous operation replays must first be enabled at the cluster level. See *Viewing Asynchronous Operation Replays* in Sybase Control Center online help.

Deploying and Managing MBO Packages

Use Sybase Control Center to deploy MBO packages created by developers to a production Unwired Server and manage package configuration. When an administrator deploys a package to a production server, they are responsible for updating package properties to use production values to replace the development or test values assigned by developers.

Properties that must be updated to production values inlclude:

- Deployment properties
- Role mappings
- Server connection properties

For details on launching the wizard and using it to reconfigure these existing development values, see *Deploying Packages* in the Sybase Control Center online help.

Multiple tasks are involved in package deployment and management. Package administration tasks vary depending on the type of package you deploy.

See also

- *Tuning Messaging* on page 168

Managing Deployed Package Subscriptions

Manage UNIFIED, messaging, and SAP Data Orchestration Engine connector (DOE-C) package subscriptions that specify the synchronization messages mobile device users receive.

Subscription management tasks include pinging, unsubscribing, recovering, suspending, resuming, resynchronizing, and logging subscriptions. Subscription tasks vary by the package type.

These subscription management tasks apply only to the package types specified in the table below. Perform each task in the Subscriptions tab of the deployed package you are managing.

Table 6. Subscription management tasks

Subscription task	Description	Summary	Package type
Ping	Ensure that push information a user provides for a device is configured correctly. If the ping is successful, notifications and subsequent data synchronizations occur as defined by each subscription. If the ping fails, open the log and check for an incorrect host name or port number.	Select the box adjacent to the device ID, and click **Ping**.	UNIFIED (replication subscriptions)
Unsubscribe	Remove a subscription from Unwired Server.	Select the box adjacent to the device ID, and click **Unsubscribe** for UNIFIED packages, messaging packages, and DOE-C packages. For Windows Mobile, the device application must include the DatabaseClass.CleanAllData(); method for data to be unsubscribed correctly. If this method is not used, **Unsubscribe** and **Subscribe** could work unpredictably.	All

Subscription task	Description	Summary	Package type
Recover	Reestablish a relationship between the device and Unwired Server. Perform recovery under severe circumstances when a device is unable to successfully synchronize data. During subscription recovery, Unwired Server purges all enterprise data on the device. It retains the device ID and subscription information so that all data can then be resynchronized and loaded onto the device.	Check the box adjacent to the subscription ID of the device, and click **Recover**.	Messaging
Suspend/resume	Control the deactivation and reactivation of package subscriptions: • Suspend – temporarily block data synchronization for a device subscribed to a particular package. • Resume – reactivate a package subscription after it has been suspended.	Select the box adjacent to the subscription ID of the device, and click either **Suspend** or **Resume**.	Messaging DOE-C
Resynchronize	Reactivate subscriptions to a deployed package. If a DOE-C subscription does not respond to the SAP DOE quickly enough, the DOE may mark that subscription's queues as "blocked" and stop sending messages to the DOE-C. Resynchronize to resume communication from the DOE to the DOE-C subscription.	Check the box adjacent to the subscription ID of the device, and click **ReSync**.	DOE-C

Subscription task	Description	Summary	Package type
Purge	Removes subscriptions that are no longer referenced by any active users.	Select the subscription, click **Purge**, and then select the criteria.	Messaging UNIFIED (replication subscriptions)

See also

- *Deploying an MBO Package to a Domain* on page 79
- *Selecting a Security Configuration for a Package* on page 77
- *Mapping Roles for a Package* on page 78
- *Managing Package Users* on page 78
- *Viewing and Editing Package Properties* on page 80

Selecting a Security Configuration for a Package

Designate a security configuration for a package in Sybase Control Center. This is a required step during package deployment, but you can later change the security configuration.

The administrator must create a security configuration in the cluster and assign it to the domain where the package is deployed before the deployer can assign the security configuration to the package.

1. In the left navigation pane, expand the **Packages** folder, and select the package to configure.
2. In the right administration pane, click the **Settings** tab.
3. Select a security configuration.

   The security profiles that appear in this list have been created by a platform administrator and assigned to the domain.
4. Click **Save**.

See also

- *Deploying an MBO Package to a Domain* on page 79
- *Managing Deployed Package Subscriptions* on page 75
- *Mapping Roles for a Package* on page 78
- *Managing Package Users* on page 78

Mapping Roles for a Package

Configure package role mapping to authorize client requests to access MBOs and operations. Domain administrators can use a role mappings table to manage logical roles at the package level.

Note: If a developer has defined a logical role, mapping is not required; the logical role is matched to the physical role of the same name and is therefore automatically mapped.

See *Roles and Mappings* in *Security* for additional information.

1. Open Sybase Control Center.
2. Select and deploy an available package.
3. Follow the wizard prompts until you reach the Configure Role Mapping page for the target package. Alternately, if you are editing package role mapping after deployment, select the **Role Mapping** tab from the **Packages** > < ***PackageName***> node in the left navigation pane.
4. Set an appropriate mapping state for each logical role. The state you choose allows you to disable logical roles (**NONE**), allow logical roles to be dynamically mapped (**AUTO)**, or manually define which physical roles must be mapped to one or more logical roles (**Map Roles**). The states of AUTO or NONE require the least administration.

See also

- *Mapping Roles for a Domain* on page 79
- *Deploying an MBO Package to a Domain* on page 79
- *Managing Deployed Package Subscriptions* on page 75
- *Selecting a Security Configuration for a Package* on page 77
- *Managing Package Users* on page 78

Managing Package Users

A package user is automatically registered when the user is successfully authenticated by the security configuration mapped to the MBO package. This is the identity used by the application to synchronize with the MBO package. The same identity is also registered as application user for native replication or messaging applications.

Package users are automatically removed when packages are removed. Those users can also be manually removed by an administrator. Typically, the administrator does the clean up to release server-side resources when the package user is no longer an active user of the MBO package. Removal of package user deletes subscription(s), server-side personalization data, cached data, and other internal artifacts associated with that user. Once the package user is removed, the device application for that user will not work until the user takes actions in the application that result in recreating the local database, synchronizing, and so on. If a user happens to be the last package user from that device, removal of that user would also result in freeing up of the device license consumed by that device.

When Unwired Server is upgraded from a previous release (such as 2.0 or 2.0.1), with MBO packages deployed to it, package users entries are automatically created for the packages on behalf of users already-registered for those packages.

See also

- *Deploying an MBO Package to a Domain* on page 79
- *Managing Deployed Package Subscriptions* on page 75
- *Selecting a Security Configuration for a Package* on page 77
- *Mapping Roles for a Package* on page 78

Deploying an MBO Package to a Domain

Deploy an MBO package to Unwired Server using Sybase Control Center. Packages are initially created by developers, but are deployed and maintained on a production Unwired Server by administrators.
In the left navigation pane, from the **Domain** node, launch the Deploy wizard.

Once an MBO package is deployed as a Replication or Messaging package, Sybase Control Center shows it as a Unified type package. The same package can be used to serve both replication and messaging device application clients. See *Deploying a Replication or Messaging Package* in Sybase Control Center online help.

Mapping Roles for a Domain

Configure role mapping in Sybase Control Center to manage logical roles and authorize client requests to access domain resources.

Prerequisites

Unwired Platform cannot query all enterprise security servers directly; to perform authentication successfully know the physical roles that are required.

Task

Typically, the role mapping used by developers are not the same as for a production system. Administrators must reset these accordingly. When you map domain-level roles, these roles are automatically applied to the packages that use the same security configuration.

1. Expand the domain-level Security node and select the security configuration to map roles for.
2. Set an appropriate mapping state for each logical role:
   - **NONE** – disable logical roles.
   - **AUTO** – allow logical roles to be dynamically mapped.

- **Map Roles** – manually map required physical roles for a logical role when physical and logical role names do not match. If names do not match, the AUTO mapping state does not work; mappings cannot occur dynamically.

The states of AUTO or NONE require the least administration.

3. To manually map roles, use the Role Mappings dialog to map a logical role to one or more physical roles. You can also map multiple logical roles to the same physical role. Add or delete available roles as needed.
   Once a logical role is manually mapped, the mapping state changes to MAPPED. Mapped roles appear in the active Physical Roles cell in the domain-wide role mappings table.

See also

- *Mapping Roles for a Package* on page 78

Viewing and Editing Package Properties

View these settings when troubleshooting, and make changes as needed to correct a problem or improve performance with a package.

1. Log in to Sybase Contol Center.
2. Expand the domain, then expand **Packages**.
3. Select the package, then select the tab for the properties you want to view or edit.
4. See the reference topics linked below for detailed information about the different settings.

See also

- *Managing Deployed Package Subscriptions* on page 75

CHAPTER 7 Mobile Workflow Package Management Overview

The goal of mobile workflow package management is to make mobile workflows available from the Unwired Server to device users. Mobile workflow package management typically requires a one-time deployment and configuration, except for ongoing package maintenance.

The mobile workflow application is a simple business process application that delivers functionality, such as sending requests and approvals through an e-mail application, to mobile device clients on supported device platforms, including Windows Mobile, iOS.

Table 7. Mobile workflow package management

Task	Frequency	Accomplish by using
Deploy mobile workflow packages	Once, unless a new version becomes available	Sybase Control Center for Unwired Platform with the Workflow node
Mobile workflow configuration that includes e-mail matching rules and context variables	Once	Sybase Control Center for Unwired Platform with the Workflow node
Device registration and user assignments to mobile workflow packages	Routine when new users or new devices are added	Sybase Control Center for Unwired Platform with the **Workflow** > **<WorkflowName>** node
Monitor users and errors	Routine	Sybase Control Center for Unwired Platform with the Monitor node

Enabling and Configuring the Notification Mailbox

Configure the notification mailbox settings that allow Unwired Server to transform e-mail messages into mobile workflows.

The notification mailbox configuration uses a listener to scan all incoming e-mail messages delivered to the particular inbox specified during configuration. When the listener identifies an e-mail message that matches the rules specified by the administrator, it sends the message as a mobile workflow to the device that matches the rule.

Note: Saving changes to the notification mailbox configuration deletes all e-mail messages from the account. Before proceeding with configuration changes, consult your e-mail administrator if you want to back up the existing messages in the configured account.

1. Log in to Sybase Control Center.
2. In the left navigation pane, click **Workflows**.
3. In the right administration pane, click **Notification Mailbox**.
4. Select **Enable**.
5. Configure these properties:
   - **Protocol** – choose between POP3 or IMAP, depending on the e-mail server used.
   - **Use SSL** – encrypt the connection between Unwired Server and the e-mail server in your environment.
   - **Server** and **Port** – configure these connection properties so Unwired Server can connect to the e-mail server in your environment. The defaults are localhost and port 110 (unencrypted) or 995 (encrypted).
   - **User name** and **Password** – configure these login properties so Unwired Server can log in with a valid e-mail user identity.
   - **Truncation limit** – specify the maximum number of characters taken from the body text of the original e-mail message, and downloaded to the client during synchronization. If the body exceeds this number of characters, the listener truncates the body text to the number of specified characters before distributing it. The default is 5000 characters.
   - **Poll seconds** – the number of seconds the listener sleeps between polls. During each poll, the listener checks the master inbox for new e-mail messages to process. The default is 60 seconds.
6. If you have added at least one distribution rule, you can click **Test** to test your configuration. If the test is successful, click **Save**.

Deploying and Managing Mobile Workflow Packages

Use Sybase Control Center to deploy mobile workflow packages created by developers, and to perform the configuration tasks required to make them available to application users on messaging devices.

Prerequisites

If a workflow is needed for a domain, then a developer must configure the context variable for a known domain. Workflows use MBO packages deployed to a specific domain. The administrator then only needs to change the context variable if the developer defined variable in the workflow is not using the same domain as a named MBO package it requies.

Task

Configuring Mobile Workflow Package Properties

Use Sybase Control Center to configure the deployed mobile workflow package general properties, matching rules, and context variables to give a mobile workflow package a different set of properties in the production environment.

1. Configure general mobile workflow properties in the **General** tab of the **Workflows > MyWorkFlow** node. General mobile workflow properties include the display name and icon for the mobile workflow package.

   See *Configuring General Mobile Workflow Properties* in Sybase Control Center online help.
2. Configure and test matching rules for the package in the **Matching Rules** tab of the **Workflows > MyWorkFlow** node. Matching rules specify how to redirect e-mail messages at runtime.

   See *Configuring Matching Rules* in Sybase Control Center online help.
3. Configure context variables for the package in the **Context Variables** tab of the **Workflows > MyWorkFlow** node. Context variables specify how to load data into the Unwired Server cache as well as the domain where the MBO package is deployed.

   See *Configuring Context Variables* in Sybase Control Center online help.

See also

- *Assigning and Unassigning Mobile Workflows* on page 83

Assigning and Unassigning Mobile Workflows

Assign mobile workflow packages to make them available to a device user. Unassign them when a package is no longer required.

1. In the left navigation pane of Sybase Contorl Center, click **Workflows > <Mobile_WorkFlow_Package>**.
2. In the right administration pane, click the **Application Connections** tab.
3. Locate the device to assign a mobile workflow package to, then:
   a) Click **Assign Workflow**.
   b) List the activation users to assign the mobile workflow package to.

      By default, no users are listed in this window. 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.
   c) Select the user or users from the list to which to assign the mobile workflow package.
   d) Click **OK**.
4. To unassign a mobile workflow package, select the User and click **Unassign Workflow**.

See also

- *Configuring Mobile Workflow Package Properties* on page 83

CHAPTER 8 DOE Package Management Overview

The goal of SAP Data Orchestration Engine (DOE) package management is to make DOE-based applications available to device users. DOE package management typically requires a one-time deployment and configuration, except for ongoing subscription management.

A package, along with its current settings for cache groups, role mappings, synchronization groups, connections, and security configuration, can be exported to an archive and imported back into Sybase Control Center for backup or to facilitate a transition from a test environment to a production environment.

Table 8. DOE package management

Task	Frequency	Accomplish by using
Deploy DOE packages	Once, unless a new version becomes available	SAP DOE Connector command line utility's **deploy** command, executed from the command prompt
Upgrade a deployed DOE package	As often as new version becomes available	
Package configuration that includes e-mail matching rules and context variables	Once	Sybase Control Center for Unwired Platform with the Packages node
Device registration and user assignments to DOE packages	Routine when new users or new devices are added	Sybase Control Center for Unwired Platform with the **Packages** > <**PackageName**> node
Monitor users and errors	Routine	Sybase Control Center for Unwired Platform with the Monitor node

Deploying a DOE Package

Use the deploy command, in the SAP DOE Command Line Utility, from the command prompt.

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 *deploy Command* 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.

See also

- *deploy Command* on page 261

Upgrading a Deployed DOE Package

To avoid conflicts and interruption in availability of the DOE package, deploy the upgraded DOE package to a different domain and let users migrate when they are ready.

1. In Sybase Control Center, create a new domain.
2. Deploy the new ESDMA bundle to the new domain.

   See *Developer Guide: DOE-based Native Applications*:

   1. "Converting the ESDMA Bundle into an Unwired Platform Package" topic.
   2. "Deploying the Unwired Platform Package" topic.

3. Ask users to switch to the new domain when they are ready.

CHAPTER 9 Mobile Data Management Overview

The goal of data management is to ensure the data tier of Unwired Platform remains stable, available, and efficient. Data management is a not a routine administration task, and for Unwired Platform, primarily involves maintaining cache data and ensuring data is delivered to the device in a timely manner, as determined by your business requirements.

Table 9. Data Management Tasks

Task	Frequency	Accomplish by using
Set up the cache group refresh schedule (to update data on the Unwired Server from the data source).	Once, unless data refreshes need to be tuned	Sybase Control Center
Create synchronization groups, and push notifications, and subscriptions to customize how updated data in the cache is delivered to the device user (varies by synchronization protocol: replication or messaging.	Periodic, as required	Sybase Control Center
Review current/historical/performance metrics	Routine	Sybase Control Center

See also

- *Backup and Recovery* on page 171

Data Mobility Configuration Dependencies

To ensure that data in the Unwired Platform mobility ecosystem remains effective and timely, administrators must carefully schedule and configure data update and delivery (synchronization) mechanisms in Sybase Control Center.

Consider that other dependencies might exist for the properties in the table below. Implement the properties to support both sides of the data mobility architecture (back-end to cache, and cache to frontline). In particular, poorly timed schedules or intervals can result in stagnant data or an unexpected user experience.

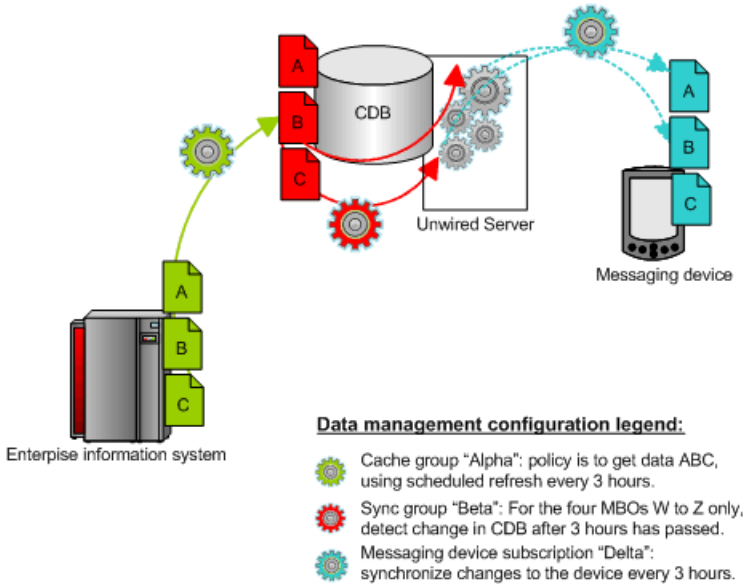
Setting	Sync type	Data updated	Usage
DCNs	Both	Server cache	The specific data units the EIS sends to Unwired Server cache. DCN updates operate outside of the cache group and cache refresh mechanisms. **Dependencies:** None. No coordination required. However, data change notifications (DCNs) do trigger the same synchronization activity on the device.
Cache group	Both	Server cache	The data the Unwired Server fetches from the EIS to update the server cache. Cache groups aggregate data updates. **Dependencies:**Design cache groups along with synchronization groups so you know which MBOs require which groups of data.
Cache refresh	Both	Server cache	Whether data is fetched from the EIS on demand or as scheduled. **Dependencies:**Coordinate with the synchronization group and device notifications (for replication only).
Subscriptions	Both	MBO data	The data required by the MBO. **Dependencies:**Design cache groups along with synchronization groups so you know which MBOs require which groups of data.
Sync group	Messaging	MBO data	The required unit of synchronization, and the frequency with which the changes are pushed to the device via the change detection interval. **Dependencies:**Coordinate the change detection interval with the cache refresh schedule used. Device push settings also influence frequency of messages are sent.
Sync group	Replication	MBO data	The required unit of synchronization. Operation replays are sent to the server (from device), and MBOs data is then downloaded to the client. **Dependencies:**The synchronization group for replication depends on the subscription, and the device notification interval.

Setting	Sync type	Data updated	Usage
Device notification	Replication	MBO data	When to trigger a synchronization by sending a notification e-mail message to the device user. **Dependencies:**Coordinate the device notification interval with the cache refresh schedule.

Message Data Flow and Dependencies

Implement the properties that determine how data flows in a messaging-based synchronization paradigm in a coordinated manner.

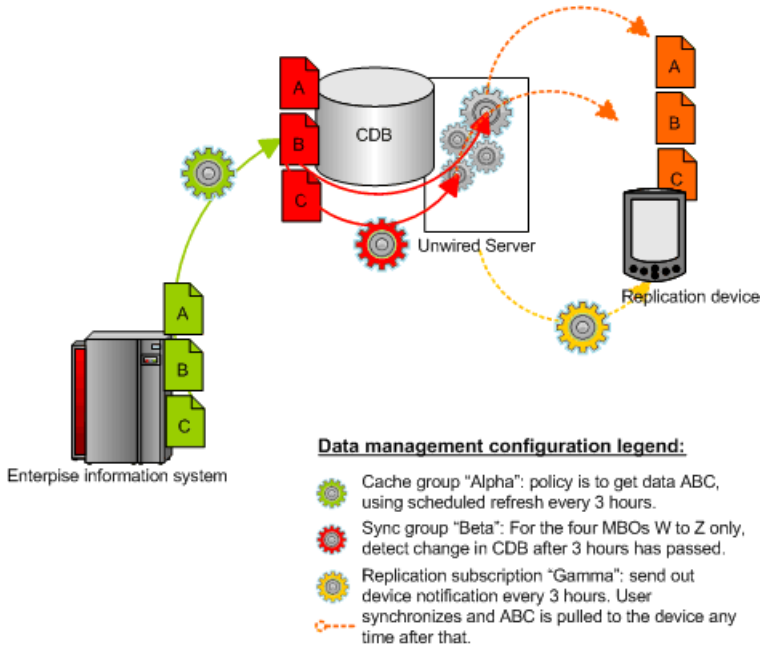
Cache refresh schedules, combined device settings and synchronization groups, work together to determine how data is disseminated to the front line from back-end EIS data sources. Coordinate these properties accordingly.



Replication Data Flow and Dependencies

Implement, in a coordinated manner, the properties that determine how data flows in a replication-based synchronization paradigm.

Cache groups, cache refresh schedules, subscriptions (in particular, synchronization groups and device notifications) work together to determine how data is disseminated to the front line from back-end EIS data sources. Coordinate these properties accordingly.



Push and Pull Synchronization with Notifications

If you require push synchronization (that is, synchronization initiated by Unwired Server), ensure that it is enabled and correctly set up to use specific gateways.

A gateway provides the interface that enables the notifier to send messages to a listener when MBO data changes in the Unwired Server cache database. Unwired Platform supports these gateways:

- HTTP – is a push-based notification for BlackBerry devices. You must manually configure this gateway.

  Note: Due to network connectivity limitations in General Packet Radio Service (GPRS)-capable Windows Mobile devices, Sybase recommends that you do not use IP number tracking and HTTP push notifications. However, because the MobiLink-based lightweight processor used for Windows Mobile devices is currently unavailable for BlackBerry, HTTP is the only gateway available for BlackBerry push. To use an HTTP gateway, you must pair the device with a BlackBerry Enterprise Server.

- DeviceTracker – is a pull-based notification for Windows Mobile devices. Uses lightweight polling using a notifier that is configured to listen for events. You do not see this gateway on the Unwired Server configuration Push Listener tab in Sybase Control Center because the gateway is configured automatically for Unwired Server. It is enabled by default and ready to use without administrator intervention.

Setting Up Lightweight Polling for a Single Client

If you do not want to set the lightweight polling configuration on Unwired Server for multiple device clients, use the client-side program for a single client.

The **poll_every** unit should be set to seconds (not minutes, hours, or a combination of the two). The lightweight poller listener on the client can be turned on/off if you do not want to receive notifications during a specific period; do not just change the interval.

1. In your program, look for where you specify the polling option right after: `...;poll_notifier=UALIGHTWEIGHT;poll_key....`
2. Change the polling option, for example: `....;poll_notifier=UALIGHTWEIGHT;poll_every=180;poll_key=.....`

Managing Deployed Package Subscriptions

Manage UNIFIED, messaging, and SAP Data Orchestration Engine connector (DOE-C) package subscriptions that specify the synchronization messages mobile device users receive.

Subscription management tasks include pinging, unsubscribing, recovering, suspending, resuming, resynchronizing, and logging subscriptions. Subscription tasks vary by the package type.

These subscription management tasks apply only to the package types specified in the table below. Perform each task in the Subscriptions tab of the deployed package you are managing.

Table 10. Subscription management tasks

Subscription task	Description	Summary	Package type
Ping	Ensure that push information a user provides for a device is configured correctly. If the ping is successful, notifications and subsequent data synchronizations occur as defined by each subscription. If the ping fails, open the log and check for an incorrect host name or port number.	Select the box adjacent to the device ID, and click **Ping**.	UNIFIED (replication subscriptions)

Subscription task	Description	Summary	Package type
Unsubscribe	Remove a subscription from Unwired Server.	Select the box adjacent to the device ID, and click **Unsubscribe** for UNIFIED packages, messaging packages, and DOE-C packages. For Windows Mobile, the device application must include the DatabaseClass.CleanAllData(); method for data to be unsubscribed correctly. If this method is not used, **Unsubscribe** and **Subscribe** could work unpredictably.	All
Recover	Reestablish a relationship between the device and Unwired Server. Perform recovery under severe circumstances when a device is unable to successfully synchronize data. During subscription recovery, Unwired Server purges all enterprise data on the device. It retains the device ID and subscription information so that all data can then be resynchronized and loaded onto the device.	Check the box adjacent to the subscription ID of the device, and click **Recover**.	Messaging
Suspend/resume	Control the deactivation and reactivation of package subscriptions: • Suspend – temporarily block data synchronization for a device subscribed to a particular package. • Resume – reactivate a package subscription after it has been suspended.	Select the box adjacent to the subscription ID of the device, and click either **Suspend** or **Resume**.	Messaging DOE-C

Subscription task	Description	Summary	Package type
Resynchronize	Reactivate subscriptions to a deployed package. If a DOE-C subscription does not respond to the SAP DOE quickly enough, the DOE may mark that subscription's queues as "blocked" and stop sending messages to the DOE-C. Resynchronize to resume communication from the DOE to the DOE-C subscription.	Check the box adjacent to the subscription ID of the device, and click **ReSync**.	DOE-C
Purge	Removes subscriptions that are no longer referenced by any active users.	Select the subscription, click **Purge**, and then select the criteria.	Messaging UNIFIED (replication subscriptions)

See also

- *Deploying an MBO Package to a Domain* on page 79
- *Selecting a Security Configuration for a Package* on page 77
- *Mapping Roles for a Package* on page 78
- *Managing Package Users* on page 78
- *Viewing and Editing Package Properties* on page 80

Cache Data Management

Managing the Unwired Server cache in the cache database ensures that enterprise data remains stable, available, and consistent across sources. Cache data management is essential to keeping data synchronized between Unwired Server and the device client application.

Cache data consists of a copy of enterprise data that is stored in a specific area of the cache database (CDB). It is used as the data repository for replication and messaging mobile business objects (MBOs) that are deployed to Unwired Server. Cache management primarily involves configuring:

- Data change notifications (DCNs) -- when data used by an application changes on an EIS server, DCNs notify Unwired Server to synchronize cache data. DCNs are useful to cache

management, since they facilitate time-sensitive data synchronization between scheduled cache refreshes.

- Cache refresh schedules -- update the cache with the most recent EIS data at regularly scheduled intervals, or on demand. The remote client database eventually retrieves updated data from the server's local copy in the CDB, using one of the supported synchronization triggers. Cache refresh schedules ensure data availability while managing the network traffic required to maintain that data.

Data Change Notifications

Data change notifications (DCNs) notify Unwired Server when data used by an application changes on an EIS server.

Developers typically use DCNs because they:

- Avoid the need to repopulate all the data in the mobile business object cache data every time data changes in the back-end EIS system.
- Allow the Unwired Server cache to be updated in real time by the EIS server, rather than administrator configured schedule.

DCNs are typically used in combination with a sychronization group. Part of the synchronization group is a change detection interval property. For messaging based sync applications, the property is used to send a "data change" message to the messaging based sync clients that subscribe to that data. Replication-based sync applications requires a subscription and the configuration of a device push notifications which then notifies the mobile clients.

DCNs and either data change messages or device notifications address these concerns:

- Users do not always know when to synchronize. Waiting for a user-triggered synchronization is unreliable.
- Using server-triggered push synchronization is reliable, but frequent synchronizations (for example, every X minutes) generates high traffic volumes for Unwired Server.
- Data consistency between the source EIS data and Unwired Server cache is limited to what administrators typically configure for a cache refresh interval for the entire system. It does not give priority to time-critical data.

You can configure either secure (HTTPS) or non-secure (HTTP) for DCNs. For details on how DCNs are created and sent see the *Developer Guide: Unwired Server*.

Cache Refreshes

Cache refreshes update cache data, either on demand, or at scheduled intervals, with the most recent enterprise information.

Cache refreshes update data for mobile business objects (MBOs) that belong to the cache group undergoing the refresh. The remote client database eventually retrieves the updated data from the server's local copy in the CDB, using one of the supported synchronization triggers.

Scheduled cache refreshes control the frequency with which objects update data and regulate network traffic by synchronizing data at strategic times, rather than pushing data changes

through as they occur. Administrators can also perform on-demand cache refreshes for cache groups at a specified time.

When a refresh occurs, the Unwired Server calls the default read operation (for each MBO in the cache group), and all of the rows that are returned from the enterprise information system (EIS) are compared to existing rows in the CDB as follows:

- If the CDB is empty, all rows are inserted.
- Unwired Server processes the row set and checks (using the primary key) whether the row already exists in the cache:
  - If it does, and all columns are the same as the EIS, nothing happens. When a client requests (by synchronizing) all rows that have changed since the last synchronization, only rows that have changed are included, which is important for performance and efficiency.
  - If the row does not exist, it is inserted and the next synchronization query retrieves the row.

Aggregate Updates to Multiple MBOs

Cache groups and synchronization groups control the synchronization schedule for cache and client MBO data.

Cache groups specify data refresh behavior for every mobile business object (MBO) within a group. MBOs can belong to only one cache group. Cache groups are created during development, at which point MBOs are grouped together based on their data refresh requirements. Since all MBOs in a cache group are refreshed simultaneously according to the group's cache policy, cache groups ensure that MBOs with related content remain consistent.

Synchronization groups are collections of MBOs that are synchronized together on client devices. MBOs within the same synchronization group can come from one or more cache groups. When a client synchronization request for a synchronization group is received, MBOs belonging to on-demand cache groups are refreshed according to the cache policy for that group.

See also

- *Schedules to Manage Update Frequency* on page 96
- *Notifications to Update Client Data* on page 97
- *Determining the Cache Interval and Schedule Refresh for a Cache Group* on page 98

Purging a Cache Group

Physically delete data that has been logically deleted from the cache. Cached data is marked as logically deleted when certain activities occur in the client application or back end.

1. In the left navigation pane of Sybase Control Center, expand the **Packages** folder and select the package to configure.
2. In the right administration pane, select the **Cache Group** tab.

3. Click **OK**.

Schedules to Manage Update Frequency

The Unwired Server cache is refreshed according to a cache policy, which defines the cache refresh behavior and properties for the MBOs within the cache group.

There are two properties that determine the cache refresh schedule, which is used with a subscription to synchronize data for mobile business objects (MBOs).

Note: You can configure a schedule refresh for a cache only if the developer specifies the cache group as a "scheduled" type during development. Otherwise, the **Schedule** tab does not appear in the Cache Properties window.

- Cache Interval – Unwired Server keeps a local copy of enterprise data in the cache database (CDB), and uses a synchronization server to manage updates between the CDB and EIS servers. When data is updated, the remote client database eventually retrieves updated data from this local copy in the CDB. The caching mechanism allows MBOs to retrieve updated data even if back-end servers fail. The cache interval balances the frequency with which the object updates enterprise data with the amount of network traffic required to maintain that data.

  A higher value for the cache may retain stale data, however, a lower value increases network traffic and may impede the client application's performance, because Unwired Server queries back-end information servers more frequently to look for changes and possibly update the CDB copy. Frequent queries typically put a higher load on the servers, as well as use more network bandwidth on the server side—the cache interval does not affect required bandwidth between the synchronization server and device client applications, nor the performance characteristics of the client applications. But the interval you choose can delay synchronization if Unwired Server must first update many records in the CDB.

  For example, if the cache interval is 0, each time a client application synchronizes, there is a pause while the Unwired Server rereads data from the EIS and updates the CDB. If, however, the cache interval is greater than 0, then the wait time depends on how long ago the data was refreshed. If the synchronization falls within a recent cache update, synchronization is almost immediate.
- Schedule Repeat – data is refreshed according to a schedule you set up. If you set up a schedule to repeatedly refresh data, information is always refreshed, regardless of the cache interval value. However, typically, you would set a schedule to closely match the cache interval, which is especially good practice for data that changes infrequently or is shared by many clients.

  However, as an administrator, you may use a scheduled repeat to look for data changes and to notify subscribed clients to synchronize when there are changes. Typically, mobile workers use an Unwired Platform client application as needed. To ensure that mobile data is relatively up-to-date, use the schedule repeat property to trigger a push notification.

See also

- *Aggregate Updates to Multiple MBOs* on page 95
- *Notifications to Update Client Data* on page 97
- *Determining the Cache Interval and Schedule Refresh for a Cache Group* on page 98

Notifications to Update Client Data

Unwired Server alerts device users to updated mobile business object (MBO) data based on synchronization group and subscription settings. A synchronization group is a collection of MBOs that are synchronized together at regular intervals. When MBOs in a synchronization group are updated, users subscribed to those synchronization groups receive either a data update or device notification, depending on the type of application they use.

Messaging synchronization clients directly receive the unit of changed data for MBOs that belong to the synchronization group. The change detection interval for the synchronization group determines how often messaging applications push MBO data to the client when cache refreshes occur.

Replication synchronization clients receive device notifications when data changes are detected for any of the MBOs in the synchronization group to which they are subscribed. Both the change detection interval of the synchronization group and the notification threshold of the subscription determine how often replication clients receive device notifications. Data updates through device notifications occur as follows:

1. Unwired Server checks the cache for data updates to MBOs according to the change detection interval configured for a synchronization group.
2. If there are data changes, the server generates device notifications.
3. Once the replication subscription notification threshold expires, Unwired Server delivers the device notifications to clients subscribed to the synchronization group.
4. Clients receive the device notifications and synchronize data for the MBOs belonging to the synchronization group.

Administrators can use subscription templates to specify the notification threshold for a particular synchronization group. They can then use these templates to create subscriptions for device users.

For replication devices, subscription settings ultimately determine when device notifications are delivered. For example, if data for a synchronization group is updated every two hours, but a device user's subscription indicates a notification threshold of three hours, Unwired Server postpones delivering these updates until the time indicated by the subscription settings.

See also

- *Aggregate Updates to Multiple MBOs* on page 95
- *Schedules to Manage Update Frequency* on page 96
- *Determining the Cache Interval and Schedule Refresh for a Cache Group* on page 98

Determining the Cache Interval and Schedule Refresh for a Cache Group

Use Sybase Control Center to create and configure cache groups for replication synchronization packages. A cache group specifies the data refresh behavior for every mobile business object (MBO) within that group.

The Unwired Server cache (also called the cache database, or CDB) is refreshed according to a cache policy, which defines the cache refresh behavior and properties for the MBOs within the cache group based on a schedule, or on demand.

1. Choose the cache group:
   a) In the left navigation pane, expand the **Packages** folder, and select the package.
   b) In the right administration pane, click the **Cache Group** tab.
   c) Select the cache and click **Properties**.
2. Choose an appropriate cache interval. The cache allows you to associate a frequency interval (hour, minute, seconds, and so on) for the cache group's refresh schedule. For example, if an interval value is set to 0 seconds, then every client synchronization repopulates the entire cache. Instead, choose a value that is longer in duration.
3. Set the repeat for the cycle of intervals.

   Sybase recommends that you choose a cache interval value that supports DCN behavior. See *Configuring Scheduled Cache Groups Properties* in Sybase Control Center online help.

See also

- *Aggregate Updates to Multiple MBOs* on page 95
- *Schedules to Manage Update Frequency* on page 96
- *Notifications to Update Client Data* on page 97

Example Data Update Models

Review these scenarios to understand different data update models you can employ.

Scenario 1: Product Sales with Expected and Unexpected Changes

Consider a mobile sales team with MBOs that interact with a shared data source for product data: a Catalog MBO, a Customer MBO, and a SalesOrders MBO.

To support this deployment the administrator might:

1. Evaluate the business context.

   Of all types of data (customer, orders, product), generally product data remains the most static: the EIS server that warehouses product data is typically updated at night. At this time, products may get added or removed, or descriptions or prices may change depending on supply and demand of those products. Regular business hours of operation are defined as 8:00 a.m. – 6:00 p.m., Monday to Friday.

2. Configure data refresh schedules based on this context.

   To accommodate the business requirements, the administrator creates a schedule repeat for the Product MBO that refreshes data daily from Monday to Friday, starting at 8:00 a.m., and a cache interval of 24 hours is used.

   The result of this configuration is that five days a week at 8:AM, the schedule *completely* reloads all Product data into the CDB, regardless of whether the CDB data is still current with respect to the 24-hour cache interval. The sales team, who knows that the data is refreshed each morning, synchronizes the Catalog MBO to update data in their local Product table.
3. Anticipate unexpected events and modify the schedule or the cache interval as required.

   Consider an incorrect price in the EIS database. If a T-shirt with a wholesale price of $10.47 is entered erroneously as $1.47, the device's Product table will be updated with the incorrect price information and must be corrected. An associate updates the EIS database, but the team must be informed of this critical data change.

   As a result, the administrator uses Sybase Control Center for Unwired Server to refresh the MBO once manually.

See also

- *Scenario 2: Urgent Alerts using Subscriptions and Schedules* on page 99

Scenario 2: Urgent Alerts using Subscriptions and Schedules

Consider an hospital's UrgentAlert MBO that has these attributes: Username, AlertTime, Message.

To support this deployment, an administrator might:

1. Evaluate the business context.

   A hospital executive demands that alerts be delivered to a specific doctor (or medical team) according to each person's unique user name, within one minute of the message being created in the EIS server.
2. Configure data refresh schedules based on this context.

   The Unwired Server administrator responds in such a way that when a new alert is entered into the back-end server, the alert is delivered to the corresponding doctor by setting the cache interval to 0 or "real time". Data is always live and immediate with no caching involved. However, the administrator also sets the schedule repeat for 1 minute and is required 24 hours a day, seven days a week. This refresh schedule is paired with a subscription template for the UrgentAlert MBO with push synchronization enabled.

When any mobile client with an application that includes the UrgentAlert MBO initially synchronizes, Unwired Server creates a push subscription. Because the administrator configured a schedule repeat for every minute, changes are delivered as they occur: when a change is detected, Unwired Server scans through the subscriptions to find all clients that are subscribed to this MBO and determines where the Username matches. When a match occurs, Unwired Server sends the client a push notification, telling doctor to run another synchronization to retrieve the new message using the UrgentAlert MBO.

See also

- *Scenario 1: Product Sales with Expected and Unexpected Changes* on page 98

CHAPTER 10 Operational Maintenance

Operational maintenance includes the regular administration activities that keep your mobile enterprise running properly.

Operational maintenance includes updating, changing, or repairing various components already deployed to an Unwired Platform deployment. Platform administrators typically perform these activities on a regular or semi-regular schedule, often during non-peak usage hours, to keep the environment running smoothly.

Monitoring, Diagnostics, and Troubleshooting

Routine monitoring of the runtime, in addition to running system diagnostics allows you to gauge the health of your mobile enterprise and troubleshoot issues that may arise. Use Sybase Control Center monitoring data, diagnostic results and runtime logs (system and domain) expressly for this purpose.

Sybase Control Center-based monitoring reduces the burden of vital, but time-consuming routine tasks, by freeing IT and administration staff to focus on other initiatives, without reducing operational health of Unwired Platform for the organization.

When used in conjunction with regular analysis of runtime logs, administrators can:

- Keep devices maintained and performing efficiently
- Keep components available and reduce failure length
- Identify and react to security or synchronization events before they impact your mobile infrastructure

System Monitoring Overview

(Not applicable to Online Data Proxy) The goal of monitoring is to provide a record of activities and performance statistics for various elements of the application. Monitoring is an ongoing administration task.

Use monitoring information to identify errors in the system and resolve them appropriately. This data can also be shared by platform and domain administrators by exporting and saving the data to a .CSV or .XML file.

The platform administrator uses Sybase Control Center to monitor various aspects of Unwired Platform. Monitoring information includes current activity, historical activity, and general performance during a specified time period. You can monitor these components:

- Security log
- Replication synchronization

- Messaging synchronization
- System messaging queue status
- Data change notifications
- Device notifications (replication)
- Package statistics (replication and messaging)
- User-related activity
- Cache activity

To enable monitoring, platform administrators must set up a monitoring database, configure a monitoring data source or create a new one, and set up monitoring database flush and purge options. By default the installer created a monitoring database, however you can use another one if you choose.

To control monitoring, platform administrators create monitoring profiles and configurations, which define the targets (domains and packages) to monitor for a configured length of time. A default monitoring profile is created for you by the installer. Monitoring data can be deleted by the platform administrator as needed.

Table 11. System monitoring tasks

Task	Frequency	Accomplished by
Create and enable monitoring profiles	One-time initial configuration with infrequent tuning as required	Sybase Control Center for Unwired Platform with the Monitoring node
Enable domain logging	One-time setup with infrequent configuration changes, usually as issues arise	Sybase Control Center for Unwired Platform with the **Domains** > <**DomainName**> > **Log** node.
Review current/historical/performance metrics	Routine	Sybase Control Center for Unwired Platform with the Monitoring node
Identify performance issues	Active	Sybase Control Center for Unwired Platform with the Monitoring node
Monitor application and user activity to check for irregularities	Active	Sybase Control Center for Unwired Platform with the Monitoring node
Troubleshoot irregularities	Infrequent	Reviewing various platform logs
Purge or export data	On demand	Sybase Control Center for Unwired Platform with the Monitoring node

Monitor

Monitor availability status, view system and performance statistics, and review system data that allow administrators to diagnose the health and security of the runtime.

Monitored operations include security, replication-based synchronization, messaging-based synchronization, messaging queue, data change notification, device notification, package, user, and cache activity. These aspects of monitoring are important to ensuring that the required data is collected.

The critical aspects of monitoring include:

1. Setting up a monitoring configuration. A monitoring configuration sets the server behavior for writing data to database, automatic purge, and data source where the monitoring data is stored.

   A default configuration is created for you, however you will likely want to customize this configuration for your environment. By default, monitoring data is flushed every 5 minutes. In development and debugging scenarios, you may need to set the flush behavior to be immediate. Set the **Number of rows** and **Batch size** properties to a low number. You can also disable flush, which results in immediately persisting changes to monitoring database. If you are setting up immediate persistence in a production environment, you may experience degraded performance. Use persistence with caution.
2. Creating a monitoring profile. A monitoring profile defines one or more domains and packages that need to be monitored.

   You can either use the **default** profile to capture monitoring data for all packages in all domains or create specific profiles as required. Otherwise, disable the **default** profile or modify it as needed.
3. Reviewing the captured data. An administrator can review monitoring data (current, historical, and performance statistics) from Sybase Control Center.

   Use the monitoring tabs to filter the data by domain, package, and time range. You can also export the data into a CSV or XML file and then use any available reporting or spreadsheet tool to analyze the data.

Monitoring Profiles

Monitoring profiles specify a monitoring schedule for a particular group of packages. These profiles let administrators collect granular data on which to base domain maintenance and configuration decisions.

A default monitoring profile is automatically created in disabled state on Unwired Server. Administrators can enable or remove the default profile, and enable one or more new monitoring profiles as required.

The same monitoring schedule can be applied to packages across different domains; similarly, you can select individual packages for a monitoring profile.

Note: Properties you configure for an Unwired Server are cluster-affecting. Therefore, to make sure they are propagated correctly, Sybase recommends that you set them only on a primary cluster server.

Planning for System Monitoring

Planning Unwired Platform performance monitoring requires performance objectives, benchmarks based on those objectives, and plans for scaling your production environment. Do this before you create your monitoring profile.

1. Understand the business requirements your system is addressing.
2. Translate those business needs into performance objectives. As business needs evolve, performance objectives must also evolve.
3. Perform capacity planning and evaluate performance of the monitoring database.

   If statistics indicate that Unwired Platform is approaching your capacity guidelines, you may want to collect this data more frequently and flush stale data. You can also use the Administration Client API to automate tasks such as exporting and purging of monitoring data.
4. Define and document a base set of statistics, which might include:
   - Peak synchronizations per hour
   - Peak synchronizations per day
   - Acceptable average response time
5. Decide how often system statistics must be monitored to create benchmarks and evaluate results for trends. Use benchmarks and trends to determine when your production environment needs to be expanded. Trend analysis should be performed on a regular basis, perhaps monthly.

Next

Open Sybase Control Center and create and configure the profiles you require to support your planning process.

Creating and Enabling a Monitoring Profile

Specify a monitoring schedule for a group of packages.

Prerequisites

Depending on the expected level of monitoring activity, ensure that the monitoring database is adequately prepared to store monitoring data.

Task

1. Open and log in to Sybase Control Center.
2. In the left navigation pane, select **Monitoring**.

3. In the right administration pane, select the **General** tab.
4. Click **New** to create a monitoring profile.
5. Enter a name for the new profile.
6. Select the **Domains and Packages** tab and choose packages to be monitored according to these options:
   - Monitor all domains and packages – select **All Domains and Packages**.
   - Monitor all packages from one or more domains – select a domain, then click **Select All Packages**. Perform this step for each domain you want to monitor.
   - Monitor specific packages from one or more domains – select a domain, then select the particular packages you want to monitor from that domain. Perform this step for each domain you want to monitor.
7. Select **View my selections** to view the packages you selected for the monitoring profile. Unselect this option to return to the package selection table.
8. Select **Enable after creation** to enable monitoring for the selected packages immediately after you create the profile. By default, this option is selected. Unselect this option to enable the monitoring profile later.
9. On the **Schedule** tab, select a schedule to specify when monitoring takes place:
   - **Always On** – this schedule requires no settings. Package activity is continually monitored.
   - **Run Once** – specify a length of time during which monitoring occurs, in either minutes or hours. Package activity is monitored for the duration specified for one time only.
   - **Custom** – specify start and end dates, start and end times, and days of the week. Package activity is monitored according to the time frame specified. See *Setting a Custom Monitoring Schedule*.
10. Click **OK**.
    A status message appears in the administration pane indicating the success or failure of profile creation. If successful, the profile appears in the monitoring profiles table.
11. To enable a profile that you did not enable during creation, select the monitoring profile and click **Enable**.

Setting a Custom Monitoring Schedule

Customize the monitoring schedule for packages within a monitoring profile in Sybase Control Center. Setting a custom schedule is the most flexible option; monitoring information is provided according to the time frame you specify.

Prerequisites

Begin creating a monitoring profile in the New Monitor Profile dialog.

Task

1. In the New Monitor Profile dialog, select the **Schedule** tab.
2. Select **Custom** as the monitoring schedule criteria.
3. To set a range to control which days the custom schedule runs, configure a start date and time, end date and time, or day of week (if applicable).
   - Select **Start Date** to set a date for when monitoring of package activity begins. To be more specific, you can also enter a **Start Time**. In this case, monitoring cannot begin until a given time on a given day has been reached.
   - Select **End Date** to set a date that ends the monitoring of package activity. To be more specific, you can also enter an **End Time**.
   - Select the days of the week that package monitoring runs. This means that for the days you select, the schedule runs every week on the day or days you specify.

   If you do not indicate a time frame, Unwired Server uses the default custom schedule, which is equivalent to Always On monitoring.
4. Click **OK**.

Configuring Monitoring Performance Properties

Configure auto-purge, flush threshold, and flush batch size settings to determine how long monitoring data is retained, and set a monitoring database to configure where data is stored.

Prerequisites

Depending on the expected level of monitoring activity, ensure that the monitoring database is adequately prepared to store monitoring data.

Task

1. In the left navigation pane of Sybase Control Center, select **Monitoring**.
2. In the right administration pane, select the **General** tab.
3. Click **Configuration**.
4. Configure auto purge settings.

   Auto purge clears obsolete data from the monitoring database once it reaches the specified threshold.

   a) Select **Enable auto purge configuration** to activate auto purge functionality.
   b) Enter the length of time (in days) to retain monitoring data before it is purged.
5. Configure flush threshold settings.

   The flush threshold indicates how often data is flushed from memory to the database. This allows you to specify the size of the data saved in memory before it is cleared. Alternately, if you do not enable a flush threshold, data is automatically written to the monitoring database as it is captured.

a) Select **Enable flush threshold configuration** to activate flush threshold functionality.
b) Select one of:
   - **Number of rows** – monitoring data that surpasses the specified number of rows is flushed from memory. Enter the desired number of rows adjacent to **Rows**. The default is 100.
   - **Time interval** – monitoring data older than the specified time interval is flushed from memory. Enter the desired duration adjacent to **Minutes**. The default is 5.
   - **Either rows or time interval** – monitoring data is flushed from memory according to whichever value is reached first: either the specified number of rows or the specified time interval. Enter the desired rows and duration adjacent to **Rows** and **Minutes**, respectively.

6. If you enabled a flush threshold, enter a **Flush batch row size** by specifying the size of each batch of data sent to the monitoring database. The row size must be a positive integer.

   The batch size divides flushed data into smaller segments, rather than saving all data together according to the flush threshold parameters. For example, if you set the flush threshold to 100 rows and the flush batch row size to 50, once 100 rows are collected in the monitoring console, the save process executes twice; data is flushed into the database in two batches of 50 rows. If the flush threshold is not enabled, the flush batch row size is implicitly 1.

   Note: By default, the monitoring database flushes data every 5 minutes. Alternatively, you can flush data immediately by removing or decreasing the default values, but doing so impacts performance and prevents you from using captured data.

7. Optional. To change the data source, select an available database from the **Monitor database endpoint** drop down list.

   Available databases are those with a JDBC server connection type created in the "default" domain. To create a new monitor database, a platform administrator must set up a database by running the appropriate configuration scripts and creating a server connection for the database in the default domain. The database server connection then appears as an option in the Monitor Database Endpoint drop down list.

8. Click **OK**.

See also

- *Monitoring Database Schema* on page 304

Monitoring Usage

Monitoring information reflects current and historical activity, and general performance during a specified time period.

Monitoring allows administrators to identify key areas of weakness or periods of high activity in the particular area they are monitoring. Access to this data helps administrators make decisions about how to better configure the application environment to achieve a higher level of performance.

The historical data is preserved in the monitor database. Performance data (KPIs for Replication, Messaging, Package Statistics, User Statistics, and Cache Statistics) for the specified time period is calculated upon request using the historical data available for that period. If monitoring data is purged for that time period, the performance data calculations will not factor in that data. It is recommended to purge monitoring data after putting in place mechanisms to export the required historical and/or performance data as needed. By default, monitoring data is automatically purged after seven days.

Also note that the processing times are calculated based on the time the request (or message) arrives on the server, and the time it took to process the request (or message) on the server. The client-side time (request origin time, and time taken to deliver to the server) are not factored into that data.

Reviewing System Monitoring Data

Review data for monitored activities in Sybase Control Center. The monitoring data is retrieved according to the specified time range. Key Performance Indicators (KPIs) are also calculated for the specified time range.

1. Open and log in to Sybase Control Center.
2. In the left navigation pane, select **Monitoring**.
3. In the right administration pane, select one of the following tabs according to the type of monitoring data you want to view:
   - **Security Log**
   - **Replication**
   - **Messaging**
   - **Queue**
   - **Data Change Notifications**
   - **Device Notifications**
   - **Package Statistics**
   - **User Statistics**
   - **Cache Statistics**

Current and Historical Data

Monitoring data is categorized as current, historical, or performance.

Current data for replication MBOs, cache, and replication packages is tracked in subtabs. Use current data to assess the state of an object and check for abnormalities: for example, whether the application is working, or if the current data request is blocked.

As the request is processed, current data is moved to historical tables and used to calculate the performance data. Use accumulated history data to derive object performance statistics: each time an activity is performed on the object, activity indicators are recorded in the table and create meaningful aggregate statistics.

Performance Data: KPIs

Performance subtabs list key performance indicators (KPIs). KPIs are monitoring metrics that use counters, activities, and time measurements, that show the health of the system. KPIs can use current data or historical data.

Metrics help administrators ascertain how close an application is to corporate performance goals. In addition, they also help you identify problem areas and bottlenecks within your application or system. Performance goal categories might include:

- System metrics related to Unwired Platform components, EIS servers, networks, and throughput of all of these elements.
- Application metrics, including counters.
- Service-level metrics, which are related to your application, such as orders per second and searches per second.

When reviewing monitoring data, administrators typically start by evaluating high-level data and may then choose to drill down into object specifics. The approach used typically depends on the goal: benchmark, diagnose, or assess system health.

KPI type	Description	Used to
Counters	Counters keep a grand total of the number of times something happens, until historical data is purged.	Monitor the system workload. Performance objectives derived from the workload are often tied to a business requirement such as a travel purchasing application that must support 100 concurrent browsing users, but only 10 concurrent purchasing users.
Time	Benchmark or assess the duration of an object or activity.	Assess the response times and latency of an object to assess service levels.
Object details	Provide summary or expanded details by object name (for example, a mobile business object, a domain, or a package). This information increases the layer of detail to counters and time values, to give context to trends suggested by the first two types of KPIs.	Analyze overall system health and troubleshoot system-wide issues.

Performance Statistics

As your production environment grows in size and complexity, perform periodic performance analysis to maintain the health of your mobility system.

The Monitoring node in Sybase Control Center is a data collection tool that helps administrators identify the causes of performance problems.

Use the Monitoring node to:

- Track application performance and identify trends
- Isolate performance problems to the resource, application, client, or user

Monitoring Data Categories

Monitoring data is organized according to object type, allowing administrators to perform focused data analysis on specific activities and Unwired Platform components. Current, historical, and performance-based statistics facilitate user support, troubleshooting, and performance tracking for individual application environments.

The replication and messaging categories are the primary sources of data relating to application environment performance. The remaining tabs present detailed monitoring data that focuses on various aspects of replication-based applications, messaging-based applications, or both.

Security Statistics

Security statistics reflect security activity for a specific monitored user. Security statistics enable you to monitor security authentication results.

Security Log Statistics

The security log reflects the authentication history of users either across the cluster, or filtered according to domain, during a specified time period. These statistics allow you to diagnose and troubleshoot connection or authentication problems on a per-user basis. Security log monitoring is always enabled.

User security data falls into these categories:

Category	Description
User	The user name
Security Configuration	The security configuration to which the device user belongs
Time	The time at which the authentication request took place
Result	The outcome of the authentication request: success or failure
Application Connection ID	The application connection ID associated with the user

Category	Description
Package	The package the user was attempting to access
Domain	The domain the user was attempting to access

Replication Statistics

Replication statistics reflect replication synchronization activity for monitored packages. Current statistics monitor the progress of real-time synchronizations, while historical statistics present data from completed synchronizations on a per-package basis. Performance monitoring uses key performance indicators to produce data about synchronization efficiency.

Through statistics that report on the duration and scope of synchronizations, as well as any errors experienced during synchronization, replication monitoring allows you to identify the rate at which synchronizations happen during specified time periods, which users synchronize data, and which mobile business objects are affected.

Current Replication Statistics

Current statistics for replication synchronization provide real-time information about in-progress synchronizations.

Unwired Server monitors replication requests using these statistical categories:

Category	Description
Application ID	The ID associated with the application.
Package	The package name.
Phase	The current synchronization activity: upload or download. During the upload phase, a client initiates operation replays to execute mobile business object (MBO) operations on the back-end system. During the download phase, a client synchronizes with Unwired Server to receive the latest changes to an MBO from the back-end system.
Entity	During the download phase, the name of the MBO with which the client is synchronizing. During the upload phase, the name of the operation that the client is performing.
Synchronization Start Time	The date and time that the synchronization request was initiated.
Domain	The domain to which the package involved in synchronization belongs.
Application Connection ID	The ID number of the connection participating in the synchronization.

Category	Description
User	The name of the user associated with the device ID.

Replication History Statistics

Historical data for replication-based synchronization consists of past synchronization details for monitored packages.

The summary view provides general information, whereas the detail view presents a more specific view of all request events during each synchronization; each row of data corresponds to a synchronization request from the client in the time frame you define:

- Click either **Details** to see more granular information on each synchronization request, or select the **Detail** option to see all synchronization request details. Detail view allows you to look at the individual messages that make up the summary view.
- Select **Summary** to see aggregated details by domain, package, and user about past synchronization events for the defined time frame.

Table 12. Detail view information

Synchronization element	Description
Application ID	The ID number associated with an application.
Package	The package name.
Application Connection ID	The ID number of the connection used in a synchronization request.
User	The user associated with the device ID.
Phase	The sync activity that occurred during this part of synchronization: upload or download. During the upload phase, a client initiates operation replays to change an MBO. During the download phase, a client synchronizes with Unwired Server to receive the latest changes to an MBO.
Entity	During download, the name of the MBO that the client is synchronizing with. During upload, the operation that the client is performing: create, update, or delete.
Total Rows Sent	The total number of rows sent during package synchronization. This data type is not supported at the MBO level.
Bytes Transferred	The amount of data transferred during the synchronization request.
Start Time	The date and time that the synchronization request was initiated.

Synchronization element	Description
Finish Time	The date and time that this part of synchronization completed.
Error	The incidence of errors during this request: true or false.
Domain	The domain to which the package involved in synchronization belongs.

Table 13. Summary view information

Category	Description
Application ID	The ID number associated with an application.
User	The name of the user associated with the device ID.
Package	The package name.
Total Rows Sent	The total number of rows sent during package synchronization.
Total Operation Replays	The total number of operation replays performed by clients during synchronization.
Total Bytes Sent	The total amount of data (in bytes) downloaded by clients from Unwired Server during synchronization.
Total Bytes Received	The total amount of data (in bytes) uploaded to Unwired Server by clients during synchronization.
Start Time	The date and time that the synchronization request was initiated.
Total Synchronization Time	The amount of time taken to complete the synchronization.
Total Errors	The total number of errors that occurred for the package during synchronization.
Domain	The domain to which the package involved in synchronization belongs.

Replication Performance Statistics

Replication performance statistics consist of key performance indicators (KPIs) that reflect the overall functioning of the application environment.

Performance monitoring highlights key totals and identifies average, minimum, and maximum values for primary activities. These calculations are dynamic, and are based on the data currently available in monitoring database for the specified time period.

All values in this table (totals, averages, maximums, minimums) apply to the specific time period you indicate:

KPI	Description
Total Distinct Package Synchronization	The total number of packages subject to synchronization.
Total Distinct Users	The total number of users who initiated synchronization requests. This value comprises only individual users, and does not count multiple synchronizations requested by the same user.
Average/Minimum/Maximum Sync Time	The average, minimum, or maximum amount of time Unwired Server took to finish a complete synchronization.
Time at Minimum/Maximum Sync Time	The time of day at which the shortest or longest synchronization completed.
Package with Minimum/Maximum Synchronization Time	The name of the package and associated MBO with the shortest or longest synchronization time.
Average/Minimum/Maximum MBO Rows Per Synchronization	The average, minimum, or maximum number of MBO rows of data that are downloaded when synchronization completes.
Average/Minimum/Maximum Operation Replays per Sync (records received)	The average, least, or greatest number of operation replays per synchronization received by Unwired Server from a client.
Total Bytes Sent	The total number of bytes downloaded by clients from Unwired Server.
Total Bytes Received	The total number of bytes uploaded from clients to Unwired Server.
Total Operation Replays	The total number of operation replays performed on the EIS.
Total Errors	The total number of errors that took place across all synchronizations.
Average/Minimum/Maximum Concurrent Users	The average, least, or greatest number of users involved in concurrent synchronizations.
Time at Minimum/Maximum Concurrent Users	The time at which the least or greatest number of users were involved in concurrent synchronizations.

Messaging Statistics

Messaging statistics report on messaging synchronization activity for monitored packages.

- Current monitoring data tracks the progress of messages from device users presently performing operation replays or synchronizing MBOs.
- Historical data reveals statistics indicating the efficiency of completed transactions.

- Performance monitoring provides an overall view of messaging payload activity intended to highlight areas of strength and weakness in the application environment.

Messaging historical data captures messages such as login, subscribe, import, suspend, resume and so on. The Import type message is a data payload message from server to client (outbound messages), while rest of the messages (login, subscribe, replay, suspend, resume) are sent from the client to server (inbound messages).

Current Messaging Statistics

Current statistics for messaging synchronization provide real-time information about in-progress synchronizations. Because messaging synchronizations progress rapidly, there is typically little pending messaging data available at any given time.

Unwired Server monitors messagin requests using these categories:

Category	Description
Application ID	The ID associated with the application.
Package	The package name.
Message Type	The type of message sent by the client to Unwired Server, indicating the current sync activity; for example, import, replay, subscribe, suspend, resume, and so on.
Entity	During the import process, the name of the mobile business object (MBO) with which the client is synchronizing. During replay, the operation that the client is performing. For all other message types, the cell is blank.
Start Time	The date and time that the initial message requesting synchronization was sent by the client.
Domain	The domain to which the package involved in synchronization belongs.
Application Connection ID	The ID number of the application participating in the synchronization.
User	The name of the user associated with the device ID.

Messaging History Statistics

Historical data for messaging synchronization consists of past synchronization details for monitored packages.

The summary view provides general information, whereas the detail view presents a more specific view of all request events during each synchronization; each row of data corresponds to a synchronization request from the client in the time frame you define:

- Click either **Details** to see more granular information on each synchronization request, or select the **Detail** option to see all synchronization request details. Detail view allows you to look at the individual messages that make up the summary view.
- Select **Summary** to see aggregated details by domain, package, and user about past synchronization events for the defined time frame.

Table 14. Detail view information

Data type	Description
Application ID	The ID number associated with an application.
Package	The package name.
Application Connection ID	The ID number of the connection that participated in the synchronization request.
User	The name of the user associated with the device ID.
Message Type	The type of message sent by the client to Unwired Server, indicating the sync activity; for example, import, replay, subscribe, suspend, resume, and so on.
Entity	During the import process, the name of the mobile business object (MBO) that the client is synchronizing with. During replay, the operation that the client is performing. For all other message types, the cell is blank.
Payload Size	The size of the message (in bytes).
Start Time	The date and time that the message for this sync request is received.
Finish Time	The date and time that the message for this sync request is processed.
Processing Time	The total amount of time between the start time and the finish time.
Error	The incidence of errors during this request; either true or false.
Domain	The domain to which the package involved in synchronization belongs.

Table 15. Summary view information

Category	Description
Application ID	The ID number associated with an application.
User	The name of the user associated with the device ID
Package	The package name

Category	Description
Total Messages Sent	The total number of messages sent by Unwired Server to clients during synchronization
Total Messages Received	The total number of messages received by Unwired Server from clients during synchronization
Total Payload Size Sent	The total amount of data (in bytes) downloaded by clients from Unwired Server during synchronization
Total Payload Size Received	The total amount of data (in bytes) uploaded to Unwired Server by clients during synchronization
Total Operation Replays	The total number of operation replays performed by clients during synchronization
Last Time In	The date and time that the last inbound request was received
Last Time Out	The date and time that the last outbound response was sent
Subscription Commands Count	The total number of subscription commands sent during synchronization; for example, subscribe, recover, suspend, and so on
Total Errors	The number of errors that occurred for the package during synchronization
Domain	The domain to which the package involved in synchronization belongs

Messaging Performance Statistics

Messaging performance statistics consist of key performance indicators (KPIs) that reflect the overall functioning of the application environment.

Performance monitoring highlights key totals and identifies average, minimum, and maximum values for primary activities. These calculations are dynamic, and are based on the data currently available in monitoring database for the specified time period.

All values in this table (totals, averages, maximums, minimums) apply to the specific time period you indicate:

KPI	Description
Total Messages	The total number of messages sent between the server and clients during synchronization.

KPI	Description
Total Distinct Devices	The total number of devices involved in synchronization. This total includes the same user multiple times if he or she has multiple devices. The value comprises individual devices, and does not count multiple synchronizations requested by the same device.
Total Distinct Users	The total number of users who initiated synchronization requests. This value comprises individual users, and does not count multiple synchronizations requested by the same user if he or she uses multiple devices.
Average/Minimum/Maximum Concurrent Users	The average, minimum, or maximum number of users involved in simultaneous synchronizations.
Time at Minimum/Maximum Concurrent Users	The time at which the greatest or least number of users were involved in concurrent synchronizations.
Average/Minimum/Maximum Processing Time	The average, minimum, or maximum amount of time Unwired Server took to respond to a sync request message.
Time at Minimum/Maximum Message Processing Time	The time of day at which the shortest or longest message processing event completed.
MBO for Maximum/Minimum Message Processing Time	The name of the package and associated mobile business object (MBO) with the shortest or longest message processing time.
Average/Minimum/Maximum Message Size	The average, smallest, or largest message sent during synchronization.
Total Inbound Messages	The total number of messages sent from clients to Unwired Server.
Total Outbound Messages	The total number of messages sent from Unwired Server to clients.
Total Operation Replays	The total number of operation replays performed by clients on MBOs.
Total Errors	The total number of errors that took place across all synchronizations.
Average/Minimum/Maximum Concurrent Users	The average, least, or greatest number of users involved in concurrent synchronizations.

Note: Reporting of KPIs related to concurrent users is based on a background task that takes a periodic snapshot of the messaging activities. Depending on the nature and length of the processing of a request, the background snapshot may not always see all the requests.

Messaging Queue Statistics

Messaging queue statistics reflect the status of various messaging queues. The data does not reveal any application-specific information, but provides a historical view of messaging

activities that communicates the efficiency of messaging-based synchronization, as well as the demands of device client users on the system.

Based on this data, administrators can calculate the appropriate inbound and outbound message queue counts for the system (configurable in the Server Configuration node of Sybase Control Center).

Messaging Queue Status

Messaging queue status data provides historical information about the processing of messaging-based synchronization requests by Unwired Server. The data indicates areas of high load and times of greatest activity. This data can help administrators decide how to handle queue congestion and other performance issues.

These key indicators monitor messaging queue status:

Statistic	Description
Name	The name of the messaging queue.
Current Queued Items	The total number of pending messages waiting to be processed by Unwired Server.
Average/Minimum/Maximum Queue Depth	The average, minimum, or maximum number of queued messages. For minimum and maximum queue depth, this value is calculated from the last server restart.
Time at Minimum/Maximum Queue Depth	The time and date at which the queue reached its minimum or maximum depth.
Type	The direction of message flow: inbound or outbound.
Total Messages	The total number of messages in the queue at one point since the last server reboot.
Bytes Received	The total number of bytes processed by the queue since the last server reboot.
Last Activity Time	The time at which the most recent message was added to the queue since the last server reboot.

Data Change Notification Statistics

Data change notification (DCN) statistics monitor notifications that are received by Unwired Server from the enterprise information server. Specifically, DCN monitoring reports which packages and sync groups are affected by notifications, and how quickly these are processed by the server.

Monitoring DCN statistics allows you to troubleshoot and diagnose performance issues if, for example, the cache is not being updated quickly enough. These statistics help to identify

which packages took longest to process data changes, as well as times of peak performance or strain on the system.

Data Change Notification History Statistics

Historical information for data change notifications (DCNs) consists of past notification details for monitored packages. Detailed data provides specific information on past notification activity for packages, and identifies which server data was affected.

Details about past notification events are organized into these categories:

Category	Description
Domain	The domain to which the package affected by the DCN belongs.
Package	The name of the package containing data changes.
MBO	The name of the MBO to which the notification applied.
Notification Time	The date and time that Unwired Server received the DCN.
Processing Time	The time that Unwired Server used to process the DCN.

Data Change Notification Performance Statistics

Data change notification (DCN) performance statistics consist of key performance indicators that reflect the efficiency of notification processing by Unwired Server.

Performance monitoring highlights key totals and identifies average, minimum, and maximum values for primary activities. These calculations are dynamic, and are based on the data currently available in monitoring database for the specified time period.

All values in this table (totals, averages, maximums, minimums) apply to the specific time period you indicate:

Key performance indicator	Description
Total Notifications	The total number of notifications sent by the enterprise information system to Unwired Server.
Average/Minimum/Maximum Processing Time	The average, minimum, or maximum amount of time Unwired Server took to process a DCN.
Time at Minimum/Maximum Message Processing Time	The time of day at which the shortest or longest DCN processing event completed.
Time of Last Notification Received	The time at which the most recent DCN was received by Unwired Server.

Key performance indicator	Description
MBO with Minimum/Maximum Notification Processing Time	The name of the package and associated mobile business object (MBO) with the shortest or longest notification processing time.

Device Notification Statistics

Device notification statistics provide data about the occurrence and frequency of notifications sent from Unwired Server to replication synchronization devices.

Historical device notification monitoring reports on the packages, synchronization groups, and devices affected by replication payload synchronization requests in a given time frame. Performance-related device notification data provides a general indication of the efficiency of notification processing and the total demand of synchronization requests on the system.

Device Notification History Statistics

Historical information for device notifications provides specific information on past device notifications, indicating which packages, synchronization groups, and devices were involved in synchronization requests.

Details about past device notification events fall into these categories:

Category	Description
Application ID	The ID associated with the application.
Domain	The domain to which the package affected by the device notification belongs.
Package	The name of the package containing data changes.
Synchronization group	The synchronization group that the package belongs to.
Application Connection ID	The ID number of the connection participating in the synchronization request.
Generation time	The date and time that Unwired Server generated the device notification.
User	The name of the user associated with the device ID.

Device Notification Performance Statistics

Device notification performance statistics provide a general indication of the efficiency of notification processing and the total demand of synchronization requests on the system.

Performance monitoring highlights key totals and identifies average, minimum, and maximum values for primary activities. These calculations are dynamic, and are based on the data currently available in monitoring database for the specified time period.

All values in this table (totals, averages, maximums, minimums) apply to the specific time period you indicate:

KPI	Description
Synchronization Group for Maximum Notifications	The synchronization group for which the maximum number of notifications were sent.
Package for Maximum Notifications	The package for which the greatest number of device notifications were sent.
Total Notifications	The total number of device notifications sent from Unwired Server to devices.
Total Distinct Users	The total number of users that received device notifications. This value comprises only individual users, and does not count multiple synchronizations requested by the same user.
Total Distinct Devices	The total number of devices that received device notifications. This is distinct from Total Distinct Users, because a single user name can be associated with multiple devices.
Enabled Subscriptions	The total number of replication subscriptions for which notifications are generated.
Time at Last Notification	The time at which the last device notification was sent by Unwired Server.
Outstanding Subscriptions	The total number of replication subscriptions, both enabled and disabled.

Package Statistics

Package statistics reflect response times for replication-based and messaging-based synchronization packages.

This type of monitoring uses key performance indicators to provide data on the efficiency of response by Unwired Server to synchronization requests. These calculations are dynamic, and are based on the data currently available in monitoring database for the specified time period.

Replication Package Statistics

Replication package statistics consist of key performance indicators (KPIs) that reflect the overall function of the application environment at the cluster or domain level. The statistics highlight key totals and identify average, minimum, and maximum values for primary activities.

These key indicators monitor replication packages:

Note: These KPIs are not applicable at the MBO level.

- Total Bytes Received
- Total Bytes Sent
- Total Operation Replays

KPI	Description
Total Devices	The total number of devices involved in synchronization. This total includes the same user multiple times if he or she has multiple devices. The value comprises individual devices, and does not count multiple synchronizations requested by the same device.
Total Rows Sent	The total number of rows sent during package synchronization.
Total Rows Received	The total number of rows received during package synchronization.
Total Errors	The total number of errors that took place across all synchronizations.
Total Bytes Received	The total number of bytes uploaded from clients to Unwired Server.
Total Bytes Sent	The total number of bytes downloaded by clients from Unwired Server.
Average/Minimum/Maximum Synchronization Time	The average, minimum, or maximum amount of time Unwired Server took to finish a complete synchronization.
Time at Minimum/Maximum Synchronization Time	The time at which the shortest or longest synchronization completed.
Total Synchronization Requests	The total number of sync requests initiated by a client.
Total Operation Replays	The total number of operation replays performed by clients on MBOs.

Messaging Package Statistics

Messaging package statistics consist of key performance indicators (KPIs) that reflect the overall function of the application environment at the cluster or domain level. The statistics highlight key totals and identify average, minimum, and maximum values for primary activities.

Note: These KPIs are not applicable at the MBO level:

- Total Subscription Commands
- Total Devices

These key indicators monitor messaging packages:

KPI	Description
Total Subscription Commands	The total number of subscription commands sent from clients to the server.
Total Devices	The total number of devices involved in synchronization. This total includes the same user multiple times if he or she has multiple devices. The value comprises individual devices, and does not count multiple synchronizations requested by the same device.
Average/Minimum/Maximum Message Processing Time	The average, minimum, or maximum amount of time Unwired Server took to respond to a synchronization request message.
Time at Minimum/Maximum Processing Time	The time at which the shortest or longest response time completed.
Total Inbound Messages	The total number of messages sent from clients to Unwired Server.
Total Outbound Messages	The total number of messages sent from Unwired Server to clients.
Total Operation Replays	The total number of operation replays performed by clients on mobile business objects (MBOs).
Total Errors	The total number of errors that took place across all synchronizations.
Total Data Push	The total amount of data transmitted from the server to clients.

User Statistics

User statistics consist of key performance indicators that reflect the overall activity of application users.

User statistics can be filtered to include users who belong to a particular security configuration. This type of monitoring highlights key totals and identifies average, minimum, and maximum values for primary user activities. These calculations are dynamic, and are based on the data currently available in monitoring database for the specified time period.

Note: These statistics are not supported for Sybase Mobile CRM and Sybase Mobile Workflow for SAP application users.

Replication User Statistics

Replication user statistics reflect the synchronization activity of a group of replication-based synchronization users belonging to a specified security configuration. These statistics include general activity-related information on a per-user basis.

These key indicators monitor replication users:

KPI	Description
Total Synchronization Requests	The total number of sync requests initiated by a client.
Total Rows Received	The total number of rows received during package synchronization.
Total Rows Sent	The total number of rows sent during package synchronization.
Total Bytes Received	The total number of bytes uploaded from clients to Unwired Server.
Total Bytes Sent	The total number of bytes downloaded by clients from Unwired Server.
Average/Minimum/Maximum Synchronization Time	The average, minimum, or maximum amount of time Unwired Server took to complete a synchronization request.
Time at Maximum/Minimum Synchronization Time	The time at which the fastest or slowest synchronization is completed.
Total Operation Replays	The total number of operation replays performed by user of mobile business objects (MBOs).
Total Errors	The total number of errors that took place across all synchronizations.
Total Devices	The total number of devices involved in synchronization. This total includes the same user multiple times if he or she has multiple devices. The value comprises individual devices, and does not count multiple synchronizations requested by the same device.

Messaging User Statistics

Messaging user statistics reflect the synchronization activity of a group of messaging-based synchronization users belonging to a specified security configuration. These statistics include general activity-related information on a per-user basis.

These key indicators monitor messaging users:

KPI	Description
Total Devices	The total number of devices involved in synchronization. This total includes the same user multiple times if he or she has multiple devices. The value comprises individual devices, and does not count multiple synchronizations requested by the same device.
Average/Minimum/Maximum Message Processing Time	The average, minimum, or maximum amount of time Unwired Server took to respond to a sync request message.

KPI	Description
Time at Minimum/Maximum Message Processing Time	The time of day at which the shortest or longest message processing event completed.
Total Inbound Messages	The total number of messages sent from clients to Unwired Server.
Total Outbound Messages	The total number of messages sent from Unwired Server to clients.
Total Operation Replays	The total number of operation replays performed by clients on mobile business objects (MBOs).
Total Errors	The total number of errors that took place across all synchronizations.
Total Subscription Commands	The total number of subscription commands sent from clients to the server.
Total Data Push	The total number of import data messages.

Cache Statistics

Cache statistics provide a granular view of cache activity eithr at the domain or package level, particularly in the areas of cache performance, mobile business object (MBO) status, and cache group status.

Cache statistics report on performance at the domain, package, MBO, and cache group levels to allow administrators to obtain different information according to the level of specificity required. These calculations are dynamic, and are based on the data currently available in monitoring database for the specified time period.

Note: These statistics are not supported for Sybase Mobile CRM and Sybase Mobile Workflow for SAP application users.

MBO Statistics

Mobile business object (MBO) status monitoring reports on cache activity at the MBO level, and thus, reflects activity for single mobile business objects.

Select **Package level MBO** to view the following key performance indicators:

Key performance indicator	Description
Cache Group	The name of the group of MBOs associated with this cache activity.
MBO	The name of the single mobile business object associated with this cache activity.
Number of Rows	The number of rows affected by the cache refresh.

Key performance indicator	Description
Cache Hits	The number of scheduled cache queries that occurred in the supplied date range.
Cache Misses	The number of on-demand cache or cache partition refreshes that occurred in the supplied date range.
Access Count	The number of cache queries that occurred in the supplied date range.
Minimum/Maximum/Average Wait Time	The minimum, maximum, or average duration of cache queries in the supplied date range. This time does not include the time required to refresh the cache in a cache "miss" scenario. Instead Minimum/Maximum/Average Full Refresh Time exposes this data.
Minimum/Maximum/Average Full Refresh Time	The minimum, maximum, or average duration of on-demand and scheduled full refresh activities in the supplied date range.

Cache Group Status Statistics

Cache group status statistics provide monitoring data about cache activity at the cache group level. The data reflects activity for all mobile business objects (MBOs) belonging to a cache group.

Select **Package level cache group** to view the following key performance indicators (KPIs):

KPI	Description
Package	The name of the package to which the associated cache group belongs
Cache Group	The name of the group of MBOs associated with the cache activity
Number of Rows	The number of rows in the cache table of the MBO
Last Full Refresh Time	The last time the cache or cache partition was fully refreshed
Last Update Time	The last time a row in the cache was updated for any reason (row-level refresh, full refresh, partitioned refresh, alternate read, or data change notification)
Last Invalidate Time	The last time the cache was invalidated

KPI	Description
Cache Coherency Window	The data validity time period for the cache group, in seconds. Can span any value in this range: • 0 shows that data is always retrieved on-demand for each client. • 2049840000 shows that the cache never expires. This occurs when you set the on-demand cache group to NEVER expire or scheduled cache group to NEVER repeat.

Refining Scope with Filters, Sorting, and Views

You can refine any view in the Sybase Contorl Center monitoring to show selected details of particular relevance.

Use these to narrow the scope of data represented in the object tabs.

1. To sort table data alphanumerically by column, click the column title. Sorting is available on all tabs of the monitoring node.
2. You can filter data by either:
   - Time – set the start and end day and time for which to show data, or,
   - Domain – check **Show Current Filter** to set the domain for which to show data (as opposed to showing all domains, which is the default). You can optionally sort these results by package name, synchronization phase, operation, and so on by selecting the corresponding option from the **Sort By** box.
3. For historical data on application tabs, you can also toggle between detail or summary views by selecting the option of the same name. Detail views show specific details of each application and each operation (update, download), whereas summaries include aggregates of each application (total messages sent, total bytes received).

Exporting Monitoring Data

Save a segment of monitoring data to a location outside of the monitoring database. Export data to back up information, particularly before purging it from the database, or to perform closer analysis of the data in a spreadsheet application.

This option is especially useful when you need to share monitoring data with other administrators and tenants. Since this task can be time-consuming, depending upon the size of the data being exported, Sybase recommends that you export the data in segments or perform the export at a time when Sybase Control Center is not in use.

Note: The time taken to export the requested data is dependent on the time range specified, and the amount of data in the monitoring database. If the export is taking too long, or the user interface is blocked for too long, another option is to export the monitoring data using the management APIs provided for exporting monitoring data. Please refer the *Developer Guide: Unwired Server Management API* for further details.

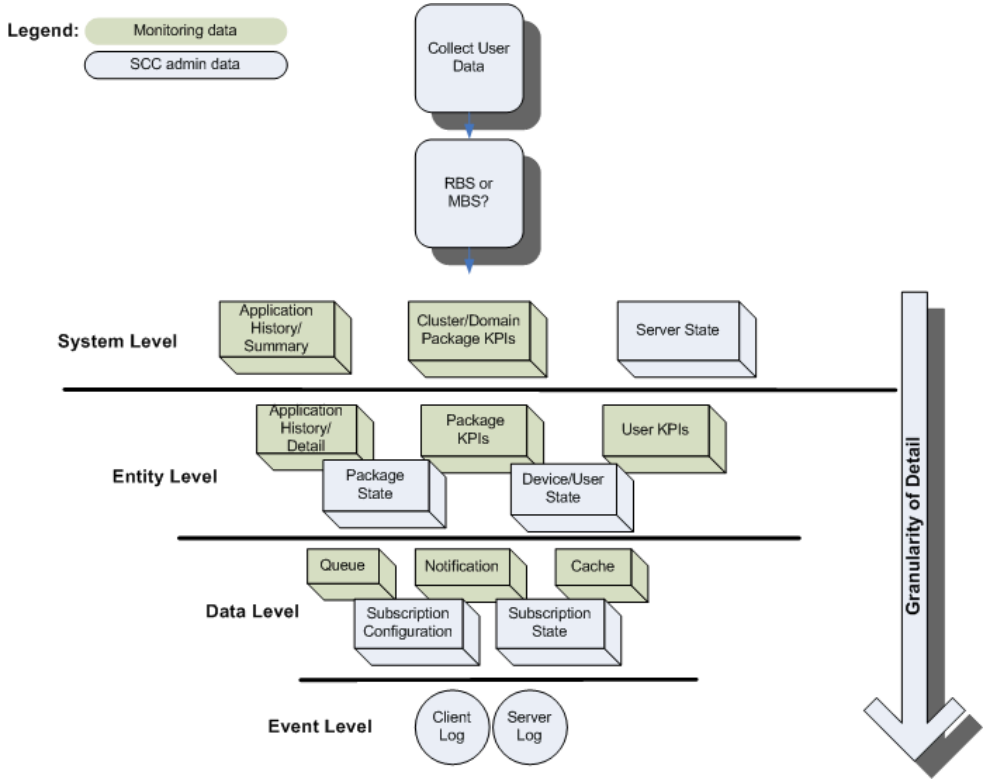
1. In the left navigation pane, select **Monitoring**.
2. In the right administration pane, select the tab corresponding to the monitoring data you want to view.
3. Perform a search using the appropriate criteria to obtain the desired monitoring data.
4. Click **Export**.
5. Select a file type for the exported data (CSV or XML), and click **Next**.
6. Click **Finish**.
7. In the file browser dialog, select a save location and enter a unique file name.
8. Click **OK**.
   All monitoring data retrieved by the search is saved to the file you specify in step 7.

Monitoring Data Analysis

Diagnosing performance issues or troubleshooting errors involves reviewing system-wide data, and typically begins with information captured by the Monitoring node in the Sybase Control Center Unwired Platform administration perspective. However, it can extend to any and all state and log data that is available to the administrator.

There is no rigidly defined troubleshooting path for administrators to take. Instead, data is reviewed and analyzed component by component until a comprehensive picture of the system emerges and gives the administrator enough symptoms to diagnose the problem.

As shown by the illustration below, monitoring and state data is collected on the platform at various levels to create an inverted pyramid of potential diagnositc data. Using this pyramid, an administrator can scale up and scale down in terms of the specificity of detail, depending on the issue being investigated. Subsequent scenarios in this section use this illustration to show you which diagnostic components may help you diagnose issues or errors.



See also

- *Device Application Performance or Issue Analysis* on page 131
- *Access Denied Analysis* on page 135
- *Data Update Failure Analysis* on page 136

Collecting Data

When diagnosing application errors and issues, the user need to provide some intial troubleshooting data that helps to identify which records and events captured by the Sybase Control Center monitoring tool should be more carefully analyzed for telling symptoms.

1. Contact the user or use an issue reporting tool that collects:
   - application type (replication or messaging)
   - and packages that make up the application
   - the user ID
   - the device ID
   - the time of the error/issue/period of interest (roughly)

2. Use the package type to start the analysis of events, as well as the package name. Other information will be necessary the more detailed your investigation becomes.

Device Application Performance or Issue Analysis

If a device user reports lagging performance or errors in a deployed application, there are a series of diagnostics the administrator can perform to investigate the problem.

Symptoms that are not revealed by the monitoring tables are likely to require a review of administrative data from other parts of Unwired Platform.

Check for	Using	See
Synchronization performance issues	Package historical data to see if high volumes or frequent operations are causing the issue or problem	*Checking Basic Application Statistics*
	Statistical information collected for the package	*Checking Package Statistics Overall*
	User statistics to see what else the user is doing at the time	*Checking User KPIs for Other Data Usage*
	Sybase Control Center server administration data to see the responsiveness of the server	*Checking Server Responsiveness in Sybase Control Center*
Unwired Server errors or failures	System logs to see if there are messages indicating whether something has failed	*Looking for Errors and Failures in SCC*

See also

- *Collecting Data* on page 130
- *Checking Package/User Histories* on page 131
- *Checking Overall Package Statistics* on page 132
- *Checking User KPIs for Other Data Usage* on page 133
- *Checking Server Responsiveness in Sybase Control Center* on page 134
- *Looking for Errors and Failures in SCC* on page 134

Checking Package/User Histories

Always begin analyzing application errors or issues by checking high-level application statistics. The goal is to see if there are any obvious outliers in typical or expected application performance. The more familiar an administrator is with the environment, the more easily the administrator can identify abnormal or unexpected behavior or performance.

If time has elapsed since an issue was reported, start by checking historical information for the entire package. Drill down into suspect rows.

1. In Sybase Control Center, click the Monitoring node, then click the tab that corresponds to the application type you are investigating, either **Replication** or **Messaging**.
2. Click **History**, then choose **Summary**, to display an aggregated history for the package.
3. Select **Show current filter**, to narrow the results. Using the troubleshooting data you gathered from the user, in the filter pane:
   a) Select the domain to which the package is deployed.
   b) Select the package name from the list of packages deployed to the domain.

   Note: In the **Domain** and **Packages** fields, you can start to type a package or domain name to narrow the list to names beginning with the characters you enter.
4. Click the **User** column to sort in alphabetical (ascending or descending) order.
5. Refine entries to those that fall within the documented time frame. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the required duration.
6. Determine whether the volumes are high for any of the cells in a row. If so, further investigate the details of that package/user pairing row.
7. On the History tab, choose **Detail view** and locate the package/user row you are investigating. This view details the synchronization request, so you can find out where in the transaction chain the problem is arising. It helps you to identify what happened at the entity level.
8. Look at these columns:

   - Total Rows Sent to see if the number of data rows being synchronized is high or low.
   - Payload size to see the number of bytes downloaded to the device for each event.

   These counters may indicate that there is a lot of data being transferred during synchronization and may require some intervention to improve performance. You can then look at the entity or phase where the problem is occurring, and whether or not there is a delay of any kind that might indicate the nature of the problem. For example, perhaps large volumes of uploaded data may be delaying the download phase. However, it may also be that this behavior is normal, and the performance lag transitory.

Next

If nothing of interest is revealed, continue by checking the package statistics for all users.

Checking Overall Package Statistics

If the package history does not reveal a high amount of volume (data or frequency of operations), proceed with evaluating the package statistics for the entire environment. Package statistics can reveal issues caused by the number of device users in the environment, and how the package is performing in environment in general.

1. In Sybase Control Center, click the Monitoring node, then click the **Package Statistics** tab.
2. Choose the type of package you want to retrieve KPIs for: messaging or replication.
3. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** so KPIs are aggregated during the specified time frame.
4. In the navigation tree, expand clusters, domains, and servers, until you locate the package to investigate.
5. Click the package name and review:

   - Total Data Push
   - Time at Minimum/Maximum/Average Synchronization
   - Total Devices

   Compare the findings to see whether or not these results are revealing. Also check the number of concurrent users so you can gauge the full range of activity (for example, use the navigation tree to scope data push results to the domain and cluster): if multiple device users are using similar packages to synchronize a similar set of data, then your system could be experiencing lags due to high demands on the server cache.

Next

If package statistics do not provide useful information about data demands, evaluate the data demands of individual users.

Checking User KPIs for Other Data Usage

If the application is not downloading large amounts of data, further investigate user-based key performance indicators (KPIs) to see if the performance issue is related to the user who is performing some other data-related action that may not be related to the package for which the issue was reported.

1. In Sybase Control Center, click the Monitoring node, then click the **User Statistics** tab.
2. Choose the type of package for which to retrieve KPIs: messaging or replication.
3. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** so KPIs are aggregated for the specified time frame.
4. If the package is deployed to a particular domain, choose the domain name you require.
5. Select the user who reported the issue.
6. Review these values :

   - Total Data Push
   - Time at Minimum/Maximum/Average Synchronization Time

   A high data push value might indicate that at some point, data demands were large. If the average synchronization and minimum synchronization times fall within normal ranges but the maximum is high, it may indicate that data volumes have created a synchronization performance lag for the package. Low data delivery for the package (determined during

package-level analysis), but high values for the individual user may suggest an unusual demand pattern which may be harmless or require further investigation. Simultaneous synchronization demands may be impacting the performance of all applications on the device.

Next

If user data demands reveal no abnormalities, continue by checking administration data in other areas of Sybase Control Center, such as Device User administrative data and subscription configuration.

Checking Server Responsiveness in Sybase Control Center

If information concerning the package, users, and the environment seem to all return normal results, you may want to investigate Unwired Server data, by checking administration data in other parts of Sybase Control Center (SCC).

1. In Sybase Control Center, select **Applications** > **Application Connections**, then use the appropriate filter to narrow the list to a specific device.
2. Verify whether data is being delivered by looking in the **Pending Items** or **Last Delivery** columns.
3. Under the Domains folder, locate the domain where the package is deployed, then expand the **Packages** node.
4. Select the package name, then choose the **Subscriptions** tab to compare the delivery results with the subscription status. Use the **Last Server Response** column to see when the last message was sent from the server to the device.

Next

If the message did not transmit and the server appears responsive, continue evaluation by *Looking for Errors and Failures in SCC*.

Looking for Errors and Failures in SCC

If data in the Monitoring node reveals nothing unusual, extend your analysis to other nodes in Sybase Control Center (SCC). Look for explicit errors or failures that may be recorded elsewhere.

This will help you determine where the error may exist: the device, the Unwired Server, or the enterprise information system (EIS) data source.

1. In Sybase Control Center, click the **Packages** node.
2. To check whether the problem exists in either the device or the EIS data source:
   a) Click the **Client Log** tab.
   b) Check the Operation and Message columns and look for any operation replays that failed with error code 500, and the reason for the failure. For example, the client log

shows all logs on device. For failed operations on the device, check the details of error message and assess what kind of error may have occurred.

3. To check whether the problem exists in either the server or the EIS data source:
   a) Expand the package tree until MBOs and operations for the package complete the tree navigation for the package.
   b) For each MBO and operation, select the node in the navigation tree, then click the **History** tab.
   c) Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the duration you require.
   d) Review the data and look for any errors and the potential causes of those errors. You can check the error history, the exception or error about the operation is listed in the details of that error. Use the message to reveal the issue and coordinate with the development team as required.

Access Denied Analysis

If a user reports an `access is denied` error, the administrator can check the security log, and from there, validate the package's security configuration.

Checking the Security Log

Validate `access is denied` messages by checking the security log.

1. In Sybase Control Center, click the Monitoring node.
2. Click **Security Log**.
3. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the specified time frame.
4. Click the **Result** column to sort rows by result type.
5. Locate any authentication failures or access denied events that are logged for the user who reported the error.
6. If you find any errors in the result column, check package names and security configurations. Then check to see if a similar result is reported for other user names. Also verify if the error persists; a heavily loaded service could cause a transient error. Transient errors are generally resolved retrying the connection.

Next

If there are no errors, investigate the security setup for the pair.

Validating Security Setup

If users are reporting `access is denied` errors where access should be allowed, validate your security setup. A security configuration is defined at the cluster level by the platform administrator, then assigned at domain and package levels by either administrator type, so it may take some analysis to determine where the problem is occurring.

Use the security log to evaluate the properties of the assigned security configuration.

1. In Sybase Control Center, expand the navigation tree in the Unwired Platform administration perspective until you locate the security configuration that generated the error. It appears either in the **Domains** > **<domain_name>** > **Security** folder or the **Security** folder at the cluster root.
2. Select the configuration you are investigating.
   - If the security configuration is assigned to a domain, validate that the role mapping is correct:
     - If the Unwired Platform user is the exact name of the user in the security repository, then no mapping is required.
     - If the Unwired Platform user differs, even slightly, then logical roles used by the package, and physical roles used in the repository must be manually mapped.
   - Review the existing security policy with the security administrator to ensure that privileges are set correctly.

Data Update Failure Analysis

If a user reports an unreceived push notification (for replication packages only), or that data appears to be updating incorrectly, administrators should follow a similar process as documented for the device application performance analysis scenario.

The adminstrator needs to first assess the synchronization performance by checking data as documented in *Device Application Performance Analysis*. From there you can specifically zero in on the device push notifications and cache statistics to see if there's an issue with how data updates are delivered (as device notifications) and configured (as subscriptions).

Checking Last Notification Timestamp

If a user reports that data is not current, you can assume that either replication-based synchronization or synchronization device notifications are not working as designed. To help you ascertain which, start by checking notifications.

Prerequisites

Before investigating notification or subscription issues, confirm that synchronization is behaving normally.

Task

1. In Sybase Control Center, click the Monitoring node, then click the **Device Notifications** tab.
2. Select **History**.
3. Click **User** to sort on user name in ascending or descending alphabetical order.
4. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the time frame you specify.
5. In the Time of Notification column, ensure that the timestamp was recently recorded, then compare the start and end time of the last known synchronization. Device notification

should always occur before a synchronization. If a notification is not received by a user, they may assume that data is not updating correctly.

Checking Cache Statistics

Cache statistics provide a granular view of cache activity, particularly in the areas of cache performance, mobile business object (MBO) status, and cache group status at different levels of use in the mobility environment.

1. In Sybase Control Center, click the Monitoring node, then click the **User Statistics** tab.
2. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** so KPIs are aggregated during the specified time frame.
3. Choose the level of cache activity to investigate. Sybase recommends that you:
   - Select **Package level cache group** to investigate cache activity for the cache group, especially if the cache group is used by multiple MBOs. Review the last refresh time to see if data has recently been refreshed from the enterprise information system (EIS), and also check to see that the affected rows are reasonable for the package.
   - Select **Package level MBO** to investigate cache activity at the MBO level. Review data related to cached rows for the MBO, including the number of cache hits and misses. The number of hits should typically be higher than the number of misses. If you do not see any hits, or see a lower number of hits than misses, the notification schedule may not working as designed. See *Validating Settings of Features that Update Data in SCC* to see how the subscription that schedules push notifications is set up.

Validating Settings of Features that Update Data in SCC

If synchronization occurs after a notification, or if a notification arrives but data is not updated, both symptoms require you to evaluate the subscription settings.

Most importantly, evaluate how the cache group interval, sync group interval, and notification threshold properties are configured. If one of these items is mistimed, the user is likely to experience an unlikely result.

1. Check the settings of the cache group used by the package.
   a) In Sybase Control Center, expand the Packages node, select the package name, then choose the **Cache Group** tab.
   b) Select the box beside the cache group name and click **Properties**.
   c) Verify:
      - The cache interval or schedule used to refresh data in the Unwired Server cache based on changes in the back-end EIS data source. Make note of that refresh interval to use in the next step.
2. Select the **Subscriptions** tab and verify:
   - That the user subscription has not been removed or suspended.

- That push synchronization is used as required. Follow up with the user to ensure that push synchronization is enabled on the device.
- That the synchronization group interval is configured appropriately based on the cache interval or schedule repeat. This value determines how frequently change detection occurs.
- That the notification threshold is configured appropriately based on the synchronization group interval. This value determines how frequently a user is notified of updated server cache data.

System Logs

Unwired Platform uses multiple logs to record events that are useful for administrators who are monitoring the environment, and maintaining the health of components.

Administrators should regularly review log messages that can be recorded at differing levels of severity.

Messages in various platform logs can provide information about:

- Configuration changes
- Successful and unsuccessful system operations (for example, deployment, synchronization and so on)
- System events and failures

Log File Locations

Use a text editor to review log files from the command line using a text editor if Sybase Control Center is not available, or to concentrate your review to a particular log.

Table 16. Log file locations

Log type	Location
Unwired Server	Aggregated Unwired Server logs: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\UnwiredServer\logs Unwired server log: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\UnwiredServer\logs\<host>_server.log Database error log: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\UnwiredServer\logs\errorlog.txt MobiLink Server error log: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\UnwiredServer\logs\mlsrv_err.log Bootstrap log: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\UnwiredServer\logs\bootstrap*.log Messaging service log details: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\UnwiredServer\logs\<Module> Device client tracing logs: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\MessagingServer\Bin\ClientTrace Mobile Workflow tracing logs: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\UnwiredServer\logs\WorkflowClient

Log type	Location
Sybase Control Center for Sybase Unwired Platform	Sybase Control Center agent log: *<UnwiredPlatform_InstallDir>*\SCC-XX\log\agent.log Gateway log: *<UnwiredPlatform_InstallDir>*\SCC-*XX*\log\gateway.log Repository log: *<UnwiredPlatform_InstallDir>*\SCC-XX\log\repository.log Request logs: *<UnwiredPlatform_InstallDir>*\SCC-XX\services\EmbeddedWebContainer\log\request-*<yyyy_mm_dd>*.log Database server log: *<UnwiredPlatform_InstallDir>*\SCC-XX\services\Repository\scc_repository.slg
Relay server and RSOE	Default log details: %temp%\ias_relay_server_host.log *%temp%* is the Windows environment variable, for example, C:\WINDOWS\Temp. **Note:** In the case of IIS Web servers, the log file is C:\WINDOWS\system32\LogFiles. However, this location can be configurable in IIS Manager. RSOE log files, by default: *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\Servers\UnwiredServer\logs*<nodeName>*.RSOE*<n>*.log For example: RBShost1.RSOE4.log.
Installer	*<UnwiredPlatform_InstallDir>* *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\InstallLogs *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\InstallLogs\silentInstall
Domain	In the domainlog database; viewable in Sybase Control Center.

Log type	Location
Cache database logs	By default, cache database errors are logged in the `errorlog.txt` file, located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs`. For a data tier installation, there are three database server logs, by default located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Data\CDB` (but may vary depending on the installation): • errorlog.txt – cache DB server log • clusterdb_errorlog.txt – cluster DB server log • monitordb_errorlog.txt – monitor and domainlog DB server log
Unwired WorkSpace log	In Unwired WorkSpace, use the **Windows** >**Show View** > **Other** > **Error Log** view, or collect the log file from the Eclipse workspace directory at `<workspace folder>/.metadata/.log file`.

Message Syntax

Unwired Platform log messages typically consist of a standard set of message elements.

- Date (year-month-day when the event occurred)
- Time (hour:minute:second when the event occurred)
- Component/module (where the event occurred)
- Severity level
- Message type (a code number associated with the severity level)
- Message text (content of the event message)

Messages may also include the administrator's login name, if the administrator performed an action.

Severity Levels and Descriptions

Unwired Platform can record messages at various severity levels.

You can control the level of messages that get recorded for Unwired Server from Sybase Control Center. See *Configuring Server Log Settings* in Sybase Control Center online help.

Log messages that typically get recorded include:

Level	Description
Debug	Detailed information useful for debugging purposes.

Level	Description
Info	General system information.
Warn	Messages about conditions that might affect the functionality of the component, such as a failure to connect to servers or authentication failures, timeouts, and successes.
Error	Messages about conditions that may require immediate attention.

Server Log

Server logs enable you to monitor system health at a highlevel, or focus in on specific issues by setting up filtering criteria using Sybase Control Center

These server logs are available:

- Unwired Server logs – collect data on Unwired Server health and performance by component, and retrieve data for all or specific searches. You can save and archive system logs, and manage log file size and rollover behavior.
- Messaging Server logs – create trace configurations for messaging modules, and retrieve trace data for all or specific messages. Export data for archive or further analysis.

Note: Properties you configure for an Unwired Server are cluster-affecting. Therefore, to make sure they are propagated correctly, Sybase recommends that you set them only on a primary cluster server.

Unwired Server Runtime Logging

Unwired Server logs collect runtime information from various embedded runtime components.

By default, all the components of the Unwired Server log are set at the INFO level, except the Other components, which are set at the WARN level. However, you can change this level as required. You should only use Sybase Control Center to set these logging values to ensure they are configured correctly. These values will be correctly transcribed to an internal file (that is, `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\Repository\logging-configuration.xml`). See *Server Logs* in Sybase Control Center online help.

You can view these Unwired Server logs in two ways:

- From Sybase Control Center – click **Servers** > ***primaryServer*** > **Log** in the left pane, and **Unwired Server** > **General**in the right pane.
  The first 150 entries initially appear in the console area, so you may need to incrementally retrieve more of the log as required, by scrolling down through the log events.
- From a text editor – browse to and open one or more of the `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs\<hostname>-server.log` files. These files may be indexed, depending on how you configure the life cycle for the servers's log file.

Configuring Unwired Server Log Settings

Configure Unwired Server log properties to specify the amount of detail that is written to the log, as well as the duration of the server log life cycle. Server log properties are only set on the primary node; property settings on secondary nodes are read-only.

How changes are applied in a cluster depends on whether you are configuring a primary or secondary server. Sybase recommends you only configure log settings on the primary server. If you change the setting on a secondary server, the configuration is updated only for that server and is temporary (eventually the primary settings are propagated to all servers in the cluster).

Additionally, you should always use Sybase Control Center to configure server logs. If you manually edit the configuration file, especially on secondary servers in a cluster, the servers may not restart correctly once shut down.

1. In the Sybase Control Center left navigation pane, click **Servers > *primaryServer* > Log**, and in the right pane click **Unwired Server > Settings**.
2. The option "Start a new server log on server restart" is set by default. When selected, this option means a new version of the log file is created after server restart, and the old one is archived.
3. Set the server log size and backup behavior that jointly determine the server log life cycle.
   a) Set the **Maximum file size**, in kilobytes, megabytes, or gigabytes, to specify the maximum size that a file can reach before a new one is created. The default is 10MB.

      Alternatively, select **No limit** to log all events in the same file, with no maximum size.
   b) Set the **Maximum backup index** to determine how many log files are backed up before the oldest file is deleted. The index number you choose must be a positive integer between 1 and 65535. The default is 10 files.

      Alternatively, select **No limit** to retain all log files.
4. For each component, choose a log level:

Component	Default Log Level
MMS	Info
PROXY	Info
MSG	Info
Security	Info
Mobilink	Info
DataServices	Info
Other	Warn
DOEC	Info

Note: DOEC only appears if you run the DOEC installer after installing SUP cluster.

Log level	Messages logged
All	Complete system information
Trace	Finer-grained informational events than debug
Debug	Very fine-grained system information, warnings, and all errors
Info	General system information, warnings, and all errors
Warn	Warnings and all errors
Error	Errors only
Console	Messages that appear in the administration console only (when Unwired Server is running in non-service mode)
Off	Do not record any messages

5. Click **Save**.
   Log messages are recorded as specified by the settings you choose. The log file is located in: `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs\<hostname>-server.log`.

Log life cycle default example

If you keep the default maximum file size and default index, an Unwired Server writes to the log file until 10MB of data has been recorded. As soon as the file exceeds this value, a new version of the log file is created (for example, the first one is `<hostname>-server.log.1`). The contents of the original log are backed up into this new file. When the `<hostname>-server.log` file again reaches its limit:

1. The contents of `<hostname>-server.log.1` are copied to `<hostname>-server.log.2`.
2. The contents of `<hostname>-server.log` are copied to `<hostname>-server.log.1`.
3. A new copy of `<hostname>-server.log` is created.

This rollover pattern continues until the backup index value is reached, with the oldest log being deleted. If the backup index is 10, then `<hostname>-server.log.10` is the file removed, and all other logs roll up to create room for the new file.

See also

- *Configuring Sybase Control Center Logging for Performance Diagnostics* on page 146
- *Configuring RSOE Logging* on page 145
- *Enabling Custom Log4j Logging* on page 148

Enabling and Disabling HTTP Request Logging for DCNs

Configure HTTP logging to record request event information logged by data change notifiications (DCNs). By default, HTTP logging for DCNs is enabled, and prints output to `<UnwiredPlatform_InstallDir>/Servers/UnwiredServer/logs/<server.name>-http.log`.

You can disable HTTP logs if you do not use DCNs.

1. Open `<UnwiredPlatform_InstallDir>/Servers/UnwiredServer/Repository/Instance/com/sybase/djc/server/ApplicationServer/${yourserver}.properties`.
2. Delete the `enableHttpRequestLog` line.
3. Save the file.
4. Restart Unwired Server.

Increasing the Maximum Post Size

Increase the size of an HTTP request to handle larger data change notifications.

The default size of an HTTP request is 10 MB. The default size is set in the `jetty-web.xml` file.

1. Stop all Sybase Unwired Platform services.
2. Open the `jetty-web.xml` file, which is located in `<UnwiredPlatform_InstallDir>\Unwired Platform\Servers\UnwiredServer\deploy\webapps\dcn\WEB-INF\`.
3. Find the property, **<Set name="maxFormContentSize" type="int">10000000</Set>** and increase the value up to 100MB, for example: **<Set name="maxFormContentSize" type="int">100000000</Set>**.
4. Save the file.
5. Restart Sybase Unwired Platform services.

Configuring RSOE Logging

By default, the Relay Server Outbound Enabler (RSOE) is configured to log only errors, which is sufficient in a production environment. Increase the log level, if you require additional detail to troubleshoot the RSOE.

You configure RSOE logging when you set up or start an RSOE. Both configuration and startup features are available on the Sybase Control Center Outbound Enabler tab by clicking **Servers** > ***ServerName*** > **Server Configuration**.

See also

- *Configuring Sybase Control Center Logging for Performance Diagnostics* on page 146
- *Configuring Unwired Server Log Settings* on page 143
- *Enabling Custom Log4j Logging* on page 148

Configuring and Enabling Relay Server Logging

By default, the relay server is configured to log only errors, which is sufficient in a production environment. Increase the log level if you require more detail to troubleshoot the relay server.

Errors appear regardless of the log level specified, and warnings appear only if the log level is greater than 0.

1. Open `rs.config`.

   The location of this file varies depending on the type of Web server used to host relay server.
2. Locate the `[options]` section.
3. Set these properties:
   - `start=no`
   - `verbosity=0`

     Edit the value to 1, 2, 3, or 4. The higher the number, the higher the degree of log event verbosity. For troubleshooting and maintenance, levels 1 or 2 should be sufficient.
4. Ensure the relay service is running, then run:

   rshost -f rs.config -u -q -qc
5. Check the relay server desktop tray icon to ensure there are no errors.
   - If there are no errors, close the command window.
   - If there are errors, check the `rs.config` file for errors and try again.
6. Validate the setup by opening the log file and confirming that the log entries reflect the log level you configure.

Configuring Sybase Control Center Logging for Performance Diagnostics

Change the logging behavior for Sybase Control Center (SCC) to better capture events for the adminstration framework.

Only enable SCC logging to diagnose performance. To enable logging:

1. Open `<SCC_HOME>\conf \log4j.properties`.
2. Edit following properties as desired:
   - `log4j.appender.agent.File`
   - `log4j.appender.agent.MaxFileSize`
   - `log4j.appender.agent.MaxBackupIndex`

If you need to diagnose SCC performance issues, review performance data with the `<SCC_HOME>\log\executionTime.log`:

1. Open *`<UnwiredPlatform_InstallDir>`*`\`*`SCC-XX`*`\plugins \com.sybase.supadminplugin\agent-plugin.xml`.

2. Add the following line to the file under the `<properties>` element:

```
<set-property property="log_MO_method_execution_time"
value="enable_log_mo_method_execution_time" />
```

3. Open *<UnwiredPlatform_InstallDir>*`\`*SCC-XX*`\conf\log4j.properties`.
4. If you are experiencing log truncation issues, edit the following lines to change the default values for maximum file size (default: 5MB) and maximum backup index (default: 10 files) to the values shown in this example:

```
## file appender (size-based rolling)
log4j.appender.executionTime=org.apache.log4j.RollingFileAppender
log4j.appender.executionTime.File=${com.sybase.ua.home}/log/
executionTime.log
log4j.appender.executionTime.layout=org.apache.log4j.PatternLayou
t
log4j.appender.executionTime.layout.ConversionPattern=%d [%-5p]
[%t] %c.%M(%L) - %m%n
log4j.appender.executionTime.MaxFileSize=50MB
log4j.appender.executionTime.MaxBackupIndex=20
## log MO method execution time
log4j.logger.com.sybase.uep.sysadmin.management.aop=INFO,executio
nTime
```

5. Restart SCC.

   The `executionTime.log` file now appears in the *<UnwiredPlatform_InstallDir>*`\`*SCC-XX*`\log` folder.

Alternately, you can use the Adobe Flex log to track performance in Sybase Control Center:

1. Modify the *<UnwiredPlatform_InstallDir>*`\`*SCC-XX*`\plugins\com.sybase.supadminplugin\agent-plugin.xml` file as indicated in step 2, above.
2. Restart SCC.
3. Log in and perform your regular administrative tasks.
4. View the execution time indicators for these operations in the cookie file `supatcookie.sol`. The location of this file varies depending on your operating system:

Operating System	Location
Windows XP	`C:\Documents and Settings\<`*username*`>\Application Data\Macromedia\Flash Player\#SharedObjects`
Windows Vista	`C:\Users\<`*username*`>\AppData\Roaming\Macromedia\Flash Player\#SharedObjects`

Operating System	Location
Macintosh OS X	`/Users/<username>/Library/Preferences/Macromedia/Flash Player/#SharedObjects`
Linux	`/home/<username>/.macromedia/Flash_Player/#SharedObjects`

5. Analyze the log using your preferred method of data analysis.

See also

- *Configuring Unwired Server Log Settings* on page 143
- *Configuring RSOE Logging* on page 145
- *Enabling Custom Log4j Logging* on page 148

Enabling Custom Log4j Logging

Use any editor (text, XML, or an IDE) to create a `log4j.xml` file.

Prerequisites

Understand the use restrictions for log4j logging.

Note: The file format of this file falls outside the scope of this document. For details about `log4j.xml` format, see *http://logging.apache.org/log4j/*.

Task

1. In an editor, create a `log4j.xml` file.
2. Define the appenders you require and save the file. Appenders must be named, have a class defined, and require one or more parameters.
3. Add the file to all servers in your cluster. The file must be saved in: `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\`.
4. Repeat these steps on each server in the cluster.
5. Restart all servers.

See also

- *Configuring Sybase Control Center Logging for Performance Diagnostics* on page 146
- *Configuring Unwired Server Log Settings* on page 143
- *Configuring RSOE Logging* on page 145

Log4j Restrictions

The default Unwired Server runtime logging and log4j logging are peer systems; the implementation of one does not usually impact the other, unless functionality overlaps.

Border conditions present in the `log4j.xml` configuration file may interfere with the configuration defined by *<UnwiredPlatform_InstallDir>*`\Servers\UnwiredServer\Repository\logging-configuration.xml`. Do not create appenders that output to:

- Any console (either System.out or System.err) with any appender, including ConsoleAppender. Log data destined to these appenders is ignored by Unwired Platform runtime components.
- The Unwired Server log. By default you can find the log file is created as *<UnwiredPlatform_InstallDir>*`\Servers\UnwiredServer\logs\<`*hostname*`>-server.log`.

Any changes you make to a `log4j.xml` configuration are applied only to the node on which the change is made; they are not propagated across the entire cluster. Therefore, you must edit the file on each node individually, as required.

Windows Event Log

Sybase Unwired Platform system messages are logged in the Microsoft Windows application event log.

Events generated by respective platform components are logged with the following source identifier values.

- Advantage
- Sybase Messaging Server
- SQLANY 12.0 (32 bit) or SQLANY64 12.0 (64-bit)
- Sybase Unwired Server
- SybaseControlCenter *X.X*

The following events are logged.:

- Server start and stop events are recorded as information events.
- Server errors such as license errors, failure to connect to runtime databases, and so forth, are logged as error events.
- Whenever a user account is locked due to repeated login failures, a message is logged as warning event.

Domain Logs

The domain log enables an administrator to monitor application activities throughout the system. Detailed views of application activities are available by subsystem. The administrator can review activities in a specific subsystem log view, view correlated data in multiple

subsystems, or view a unified log across all subsystems. The administrator must enable logging, and then use log filters to view data of interest.

By default, only error messages are recorded in each domain's log. To enable domain logging, you must create a log profile. See *Creating and Enabling Log Profiles* in *Sybase Control Center* online help.

Supported Log Subsystems

Log subsystems provide categories that enable you to filter and correlate application data at a more granular level. Understanding these subsystems enables you to formulate more specific filters for tracking application activities throughout the system.

Subsystem	Description
All	Provides a unified view of all subsystems, enabling you to look for activities, trends, and symptoms across multiple subsystems.
Synchronization	Provides a view of synchronization activities. Within this subsystem, additional categories include data synchronization, operation replay, subscription, result checker, cache refresh, and data services (DS) interface.
Device Notification	Provides a view of device notification activities.
DCN	Provides a view of data change notification (DCN) activities. Within this subsystem, additional categories include general DCN, and workflow DCN.
Security	Provides a view of security-related activities.
Error	Provides a view of errors.
Connection	Provides a view of connection activities. Within this subsystem, additional categories include DOE, JDBC, RES, SAP®, and SOAP connections.
Proxy	Provides a view of Online Data Proxy connection-related activities. Within this subsystem, categories include request response, and push.

Managing Domain Logs

Configure settings to perform logging for your applications whenever needed.

Messages are logged for synchronization, data change notification, device notification, package, user, and cache activities among others. The following aspects of logging are important to ensure that performance of the applications is minimally impacted and the required data is collected. In production system, it may be used to diagnose application issues. A developer may also use it in a development or test environment for debugging purposes.

The critical steps for configuring the domain log include:

1. Set up the proper domain log configuration. A domain log configuration sets the server behavior for writing data to database, automatic purge, and data source where the log data is stored. A default configuration is created for you, however you will likely want to customize this configuration for your environment.

   By default, log data is flushed every 5 minutes. In development and debugging scenarios, you may need to set the flush behavior to be immediate. Set the Number of rows and Batch size properties to a low number. You can also disable flush, which results in immediately persisting changes to the database. If you are setting up immediate persistence in a production environment, you may experience degraded performance. Use persistence with caution. This configuration can be done by platform administrator only.
2. Create a logging profile. A logging profile allows a platform administrator or domain administrator to define the target of logging. Unwired Server can be configured to log messages for a specific application or package, security configuration or user, device connection, and/or backend connection type or specific one, or a combination thereof.
3. Review the captured data. A platform or domain administrator can review logged data using log filters. The filters enable you to retrieve data logged for a specific thread, application, user, connection, etc. among other options.

All of the above steps are performed in Sybase Control Center. Refer to the topic group *Domain Log* in *Sybase Control Center* online help for details.

Planning for Domain Logging

Domain logging is an option for obtaining detailed information on all aspects of device, user, application, domain, and data synchronization related activities. The capability can have an adverse influence on system performance, so planning is important before using it on a production system.

Follow these guidelines for implementing domain logging:

1. Understand the business requirements that your logging needs to address. Typical usage of the domain logging data would be for debugging application issues whenever they are reported.
2. Translate those business needs into logging objectives (how frequently it may be used, the duration of logging, amount of generated data, life span of generated log, and so forth). As business needs evolve, the logging configuration must also evolve.
3. To isolate the performance impact and meet your capacity planning outcome, you may run the monitor and domain log database on a separate host (from Cache DB and Cluster DB host).

   See *Isolating the Monitoring and Domain Logging Databases from Cache and Messaging Databases*.
4. Identify the target of logging for tracking requests through the system. The logging system offers flexibility to enable logging at multiple levels of granularity, and therefore it is of utmost importance to enable logging at proper granularity level so that the necessary data

is collected without incurring performance or resulting in major log data volume. Logging activities for the target data result in a major load on the system.

5. Perform some tests with simulated application loads, to evaluate performance of the domain logging database.

See also

• *Isolating the Monitoring and Domain Logging Databases from Cache and Messaging Databases* on page 23

Enabling and Configuring Domain Logging

Configure auto-purge, flush threshold, and flush batch size settings to determine how long domain log data is retained, how frequently it is written to database from server nodes, and set a domain log database connection to configure where domain log data is stored.

1. In the left navigation pane of Sybase Control Center, select **Domains**.
2. Under the domain node, select **Log**.
3. In the right administration pane, select the **Settings** tab. These settings are used for all domains.
4. Click **Configuration**.
5. Configure auto purge settings.

   Auto purge clears obsolete data from the database once it reaches the specified threshold.

   a) Select **Enable auto purge configuration** to activate auto purge functionality.
   b) Enter the length of time (in days) to retain monitoring data before it is purged.
6. Configure flush threshold settings:

   The flush threshold indicates how often data is flushed from memory to the database. This allows you to specify the size of the data saved in memory before it is cleared. Alternately, if you do not enable a flush threshold, data is immediately written to the domain log database as it is captured.

   a) Select **Enable flush threshold configuration** to activate flush threshold functionality.

   Note: Enabling flush configuration is a good practice for performance considerations. Be aware there may be a consequent delay in viewing data, until data is stored in the database.

   b) Select one of:
      - **Number of rows** – domain log data that surpasses the specified number of rows is flushed from memory. Enter the desired number of rows adjacent to **Rows**. Disabled by default.
      - **Time interval** – domain log data older than the specified time interval is flushed from memory. Enter the desired duration adjacent to **Minutes**. The default is 5.
      - **Either rows or time interval** – domain log data is flushed from memory according to whichever value is reached first: either the specified number of rows or the

specified time interval. Enter the desired rows and duration adjacent to **Rows** and **Minutes**, respectively.

7. If you enabled a flush threshold, enter a **Flush batch row size** by specifying the size of each batch of data sent to the domain log database. The row size must be a positive integer.

   The batch size divides flushed data into smaller segments, rather than saving all data together according to the flush threshold parameters. For example, if you set the flush threshold to 100 rows and the flush batch row size to 50, once 100 rows are collected in the console, the save process executes twice; data is flushed into the database in two batches of 50 rows. If the flush threshold is not enabled, the flush batch row size is implicitly 1.

   Note: By default, the domain log database flushes data every 5 minutes. Alternatively, you can flush data immediately by removing or decreasing the default values, but doing so impacts performance.

8. Optional. To change the data source, select an available database from the **Domain log database endpoint** drop down list.

   Available databases are those with a JDBC server connection type (SQL Anywhere) created in the default domain. To create a new database, a platform administrator must set up a database by running the appropriate configuration scripts and creating a server connection for the database in the default domain. The database server connection then appears as an option in the Domain Log Database Endpoint drop down list.

9. Click **OK**.

Reviewing Domain Log Data

An administrator reviews logged data by creating log filters. The filters enable you to retrieve data logged for a specific thread, application, user, connection, among other options.

You can retrieve log data without using any filters, however, when there is large number of activities being logged, it may be advisable to filter the results to a more manageable size by specifying search conditions in the log filter (user, application, or thread-id).

You can combine multiple log filters that are common with sub-system specific filters when viewing in a sub-system view, and combine multiple sub-system filters in the ALL tab to retrieve the data of interest.

Creating Log Filters

Filter the log data by creating filters across subsystems that define the appropriate search criteria.

1. In the left navigation pane of the Sybase Control Center, select the**Domains** node.
2. Select the domain node and select the **Log** node.
3. In the right administration pane, select the **General** tab.
4. Select **+** to add a filter definition to a subsystem.
5. In the Filter Definition dialog, enter the **Name** and **Description** of the filter.

6. Select the **Sub System**.
7. Select the filter criteria and assign values to the criteria selected. You can use the logical operations to compose the criteria.

   Note: You use the 'AND' logical operator to highlight filter relations belonging to the same subsystem. Filter definitions among multiple subsystems use the 'OR' logical operator.
8. Click **OK**.

Reusable Log Filters

Create reusable log filters that you can use as a base. One strategy is to create a base log filter for each of the supported log subsystems, and for significant categories within subsystems. Another strategy is to create common log filters (useful across subsystems) on specific criteria, such as thread ID, user, package, and so forth.

You can modify these base log filters as needed for more specific searches, or clone the log filter and modify it for a specific search.

Synchronization Performance Tuning

Whether you use the replication or messaging payload protocol, tuning synchronization provides the highest throughput while limiting CPU and memory consumption to reasonable operational levels. You can refine performance an ongoing capacity, or when new applications are deployed, or if there are changes to existing application functionality, load, and so on.

Tuning recommendations vary depending the payload protocol.

Performance Considerations for Replication Payloads

Understand fundamentals of entity-state replication, so you can understand how to improve performance for this protocol.

Entity-state replication has two distinct operation and data transfer phases where only differences are transferred: upload (where only updates from the mobile client to the server are integrated with data in the mobile middleware cache and pushed to the EIS) and download (where EIS delta changes are determined for, and supplied to, the specific client). These phases are conducted as discrete transactions to protect the integrity of synchronization. Synchronizations are carried out within a single data transfer session, allowing for exchanges of large payloads. Therefore the goal of tuning performance for the replication payload is to ensure those transfers of large volumes are handled effectively and not to create bottlenecks in the runtime.

Considerations that affect performance can be separated into synchronization phases and architecture components.

Entity-state replication use cases center around three primary phases in moving data between server and client, each of which need to be considered when choosing performance setting

values. Each phase describes mobility environment components, and how they affect performance: MBO development and data design, EIS data models , and Unwired Server runtimes. These phases are:

- Initial synchronization – where data is moved from the back-end enterprise system, through the mobile middleware, and finally onto the device storage. This phase represents the point where a new device is put into service or data is reinitialized, and therefore represents the largest movement of data of all the scenarios. In these performance test scenarios, the device-side file may be purged to represent a fresh synchronization, or preserved to represent specific cache refreshes on the server.

  For this phase, the most important performance consideration is the data and how it's partitioned (EIS design), and loaded by the MBO (MBO development) using operation parameters, synchronization filter parameters). Synchronization parameters determine what data is relevant to a device; they are used as filters within the server-side cache. Load parameters determine what data to load from the EIS into the Unwired Platform cache.
- Incremental synchronization – involves create, update, and delete (CUD) operations on the device where some data is already populated on the device and changes are made to that data which then need to be reconciled with the cache and the back-end enterprise system. When create and update operations occur, changes may be pushed through the cache to the back end, reads may occur to properly represent the system of record, for example, data on the back end may be reformatted, or both. This scenario represents incremental changes to and from the system of record.

  As in the initial synchronization phase, the EIS accounts for the bulk of the device synchronization response time: a slow EIS consumes resources in the Unwired Server, with the potential to further impede devices that are competing for resources in connection pools. Additionally, the complexity of the mobile model, measured by the number of relationships between MBOs, has a significant impact on create, update, and delete operation performance. Shared partitions among users or complex locking scenarios involving EIS operations can become a major performance concern during device update operations. Cache and EIS updates are accomplished within the scope of a single transaction, so other users in the same partition are locked out during an update of that partition. Consider denormalizing the model if complex relationships cause performance concerns.
- Data change notification (DCN) – changes to the back-end data are pushed to the mobile middleware and then reconciled with the device data. DCN is typically observed in the context of additional changes to the device data so that changes from both device and back end are simultaneously impacting the mobile middleware cache.

  DCN efficiently updates the Unwired Server because it does not require the Unwired Server to poll the EIS or to refresh the cache based on a schedule. EIS DCN applied to the cache is independent of the client synchronizations. If DCN data is located in a shared partition, multiple devices benefit from the single EIS update to the cache. There are several ways to materially improve DCN performance:

- Use a load-balancer between the EIS and the cache – DCNs can be efficiently applied across an Unwired Platform cluster, as each node in the cluster helps to parse incoming payloads.
- Combine multiple updates into a single batch.
- Run DCNs from a multithreaded source to parallelize updates. Note that there is a diminishing return beyond three to four clients, in large part due to the nature of the model.

Different models exhibit different performance characteristics when applying updates, so proper analysis of application behavior is important.

See also

- *Performance Considerations for Messaging* on page 162

Overview of Replication Tuning Recommendations

Replication tuning recommendations are designed to maximize bandwidth between the Relay Server and its Outbound Enablers, Unwired Server, and the EIS connections.

Sybase recommendations touch on components outside and inside the LAN:

- For large replication payloads greater than 4MB, increase the bandwidth between the Relay Server and the Unwired Platform cluster nodes by increasing the number of Relay Server Outbound Enablers (RSOEs) and increasing the shared memory of the relay servers.
- Ensure adequate processing bandwidth for peak conditions by setting an adequate replication thread count for the entire cluster (the combination of each node's replication thread count) and JVM memory settings you configure for Unwired Server. The replication thread count is the point where Unwired Server and cache database (CDB) throttling, or limiting, occurs.

  The speed of the CDB storage and storage controllers is the single most important factor in providing good system performance. Staging of mobile data is performed within the CDB in a persistent manner such that if a device synchronizes. The CDB database server itself is largely self-tuning, although typical database maintenance is essential for proper performance.
- Provide as many resources as are available as you work back to the EIS. Do not limit connection pools.

Relay Server and Outbound Enabler Performance Tuning Reference

Use this table to quickly ascertain which Relay Server and Outbound configuration properties you can adjust to tune performance.

Table 17. Recommendations Generally Suitable for 64-bit Production Servers

Component	Function	Default	Production Recommendation
Relay Server shared memory	Settings for shared memory buffer. Remember that shared memory must account for concurrent connections, especially with large payloads.	10MB	2GB
Outbound Enable r deployment	With SQL Anywhere®, each RSOE provides an upload and download channel to the relay server.	1 Outbound Enabler per server	3 Outbound Enablers per synchronization (replication or messaging) port, especially when initial synchronizations are larger than 4MB

Unwired Server Replication Tuning Reference

Use this table to quickly ascertain which runtime properties you can adjust to tune replication performance.

Table 18. Recommendations Generally Suitable for 64-bit Production Servers

Property	Function	Default	Production Recommendation
Thread count	Controls the concurrent number of threads that service devices for synchronization. This setting controls the amount of CPU used by the Unwired Platform and CDB tiers. If the processor of either the CDB or the Unwired Server is excessively high, you can use this setting to throttle the number of requests and limit contention for resources. Low settings decrease parallelism; high settings may cause undue contention for resources.	10 per server	For a 2-node cluster, use 12 for each node. For a single node, use 24.

Property	Function	Default	Production Recommendation
Synchronization cache size	The maximum amount of memory the server uses for holding table data related to device users, network buffers, cached download data, and other structures used for synchronization. When the server has more data than can be held in this memory pool, the data is stored on disk.	50MB	1GB
JVM minimum heap size	The minimum memory allocated to the differencing and cache management functions of the server.	64MB	Sybase recommends that the value not exceed 700MB, or Unwired Server is unable to start.
JVM maximum heap size	The maximum memory allocated to the differencing and cache management functions of the server. Choose the heap size carefully, otherwise you may have issues with Unwired Server startup. Choose a JVM setting that is appropriate for your: • Server hardware configuration (for example, 64 bit/32 bit, RAM size, VM size). • Size of objects and encoding. Because Unwired Server uses base64 as default binary encoding, it causes a 3-factor growth from the original size. For example, 1M after encoding to base64, uses around 3M memory. For multiple concurrent users, the memory would be multiplied. So we need to calculate the maximum heap size based of server based on above information.	256MB	For a 64-bit operating system, Sybase recommends 1 gigabyte for a normal configuration, but 2 gigabyte for a stress configuration (which can vary depending on what RAM is available).

Note: The synchronization differencing algorithms are a key feature of replication; this technology runs in the JVM. You must provide adequate memory to these components. If these algorithms are memory starved, the JVM spends an inordinate amount of time garbage collecting memory, and synchronizations back up in the internal queues. You can monitor

process memory usage with tools like SysInternal's Process Explorer to determine the actual amount of memory in use by Unwired Platform, and adjust the JVM heap size accordingly

Cache Database Performance Tuning Reference

Use this table to quickly ascertain which cache database (CDB) properties you can adjust to tune performance.

Because the replication payload protocol uses entity-state replication and a differencing algorithm to determine what to synchronize to each device, the CDB is one of the most critical components for performance in terms of processing power, memory, and disk performance. The CDB must also be scaled vertically (on larger hardware) because it supports all of the nodes of the cluster.

Table 19. Recommendations Generally Suitable for 64-bit Production Servers

Property	Function	Default	Production Recommendation
Thread count	Limits the number of tasks (both user and system requests) that the database server can execute concurrently. If the database server receives an additional request while at this limit, the new request must wait until an executing task completes.	20	200
Initial cache size	Sets the initial memory reserved for caching database pages and other server information. The more cache memory that can be given the server, the better its performance.	24MB	8GB
Disk configuration	Isolate the data and log on distinct physical disks. Of the two files, the log receives the most activity. Use disk controllers and SAN infrastructure with write-ahead caching in addition to high-speed disk spindles or static memory disks. The same database must support all cluster members. The faster these database drives perform, the better the performance on the entire cluster.	1 disk	2 disks 10K RPM

Property	Function	Default	Production Recommendation
Connection pool maximum size	Sets the maximum (JDBC) connection pool size for connecting to the CDB from each cluster member. In the example production configuration, each Unwired Server is allowed more connections than the number of threads set in the CDB thread configuration. This ratio helps to ensure that the CDB database server is the control point for limiting resource contention in the cluster. In a server configured with the CDB connection pool settings set to 0, the actual number of connections used will correspond to the number of Unwired Server threads in use. The number of internal connections in use at any one time per thread varies, although fewer than four is typical.	100	0 (unlimited)

EIS Connection Performance Tuning Reference

Use this topic to quickly ascertain which EIS connectivity properties you can adjust to tune performance.

Property	Function	Default	Production Recommendation
Connection pool maximum size	Ensure that adequate EIS resources are allocated for servicing the mobile infrastructure. The actual number of connections necessary varies, based on the maximum number of Unwired Platform threads in use at any time and the duration it takes for the EIS to respond. If possible, allocate a connection for each thread (or leave the setting unbounded). If you must limit the number of EIS connections in the pool to a lower number than the number of Unwired Platform threads and you experience timeouts, you may need to adjust the EIS connection timeout values ; however, these connections will impede other threads competing for EIS connections.	Varies, depending on type: JDBC is 10, Proxy is 25, and Web Services, SAP DOE, and SAP JCo have no limit by default.	0 (unlimited)

Tuning Replication

Tune your production environment after all components have been deployed and started.

1. Isolate the monitoring and domain logging databases from the cache server.

   This isolation only needed if you are using monitoring and domain logging in the production system, and want to reduce any database or storage contention. See *Isolating the Monitoring and Domain Logging Databases from Cache and Messaging Databases*.
2. Disable all enabled monitoring profiles that currently exist.

   Normally, monitoring in a cluster configuration occurs at two levels — the cluster and the domain. Sybase recommends that you turn off monitoring when assessing performance.

   a) In navigation pane of Sybase Control Center, click **Monitoring**.
   b) In the administration pane, select the profile name and click **Disable**.
   c) Validate that all monitoring is disabled by opening `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\monitoring\MonitoringConfiguration\domainLogging.properties`, and verifying that “status”, “enabled”, and “autoStart” are set to false.

   See *Monitoring Profiles* in the Sybase Control Center online help.
3. In Sybase Control Center, stop all Unwired Servers in all clusters.
4. Stop the cache service on the data tier node.
5. Use the dblog utility to isolate the cache disk and log on to fast storage.

   For details, see *Using a Different Database and Log Path*.
6. Set the new cache thread count, pool size, and initial cache size.

   See *Changing the Cache Database Server Thread Count and Pool Size*.
7. Restart the cache service and all Unwired Servers.
8. In Sybase Control Center, adjust the EIS connection pools to high values:

   a) In navigation pane of Sybase Control Center, click **Connections**.
   b) In the administration pane, set the Max Pool Size property for each EIS connection type.

      If the Max Pool Size property does not appear for a connection, you must manually add it.
9. In Sybase Control Center, update the Unwired Server replication synchronization properties on the primary server.

   a) In the left navigation pane, click **Servers**>**SeverName**>**Server Configuration**.
   b) In the right administration pane, click the **Replication** tab.
10. In Sybase Control Center, set thread stack size and JVM heap sizes for the server.

    a) In the left navigation pane, click **Servers**>**SeverName**>**Server Configuration**.

b) In the right administration pane, click the **General** tab then click **Performance Configuration**.
c) Adjust the thread stack size and JVM heap sizes to the new values.
d) Repeat for all servers.

11. Restart all Unwired Servers.

Testing CPU Loads for Replication

Perform a mixed-load test to ascertain whether the throughput to the cache and EIS is sufficient. Do this by testing then observing CPU load trends.

1. Run a typical application maximum mixed-load test on the Unwired Server (5% initial sync, 95% subsequent sync) and observe the cache CPU load.

Do not run this mixed-load test against Relay Server; you want to first ascertain the maximum throughput of just the Unwired Server and cache.

2. If the cache CPU load is too high, too many threads could be impacting performance:
a) Decrease the replication thread count on each server in Sybase Control Center.
b) Restart all servers.
c) Observe the cache CPU load again.

3. If the cache CPU load is low, fine tune the replication thread count until one yields the maximum through put. Check this by:
a) Note the client response times of other synchronization types and try increasing the replication thread count in small increments.

Other types of synchronization include initial to initial, subsequent to subsequent, similar payload, cache policy, and so on.

b) Repeat this process until the synchronization response times are as low as possible for the same number of clients for all types of synchronizations.

Once the relative client response time is as low as possible without further increasing the replication thread count, you have reached the configuration that yields the maximum throughput. In other words, tune the thread count to yield the best overall response times. Usually, this thread count number is small.

4. Repeat the mixed-load test on EIS backends.

This checks for EIS latencies and cache policies, both of which can change the performance of the server cluster.

5. Introduce relay server into the environment, and repeat the mixed-load test.

6. Once the environment has stabilized and tuned, enable on and configure only the monitoring profiles you need.

Performance Considerations for Messaging

Understand fundamentals of messaging so you can understand how to improve performance for this payload protocol.

With messaging payloads there is a trade off between efficiency and immediacy. Some messages arrive earlier than in entity-state replication because the payload is spread out over

many relatively small messages that are delivered individually, as opposed to being delivered as a single large download. Therefore the goal of tuning messaging performance is to perform multiple transfers of small payload volumes quickly, in addition to keeping changes in the correct order.

The messaging use cases center around three primary phases in moving data between server and client, each of which need to be considered when choosing performance setting values. Each phase describes mobility environment components, and how they affect performance: MBO development and data design, EIS data models and runtime servers, and Unwired Server runtimes. These phases are:

- Initial subscription – A mobile application subscribes to an messaging package to receive data as "import" messages from the server. Upon the receipt of the subscription request from the device, the server checks security and executes a query against the cache database (CDB) to retrieve the data set, which it then turns into a series of import messages to be sent to the device. The maximum message size is currently fixed at 20KB. Increasing the maximum allowable size would allow the same amount of import data with fewer messages. However, in some devices, large message size can trigger resource exhaustion and failures.

  The subscription phase is the most expensive or time consuming portion of the life cycle, especially for a large initial data set. However, compared to entity-state replication payloads, messaging payloads can take longer to populate the data on the device database because of the aggregation of fine-grained messages. This latency is due to the additional cost of providing reliability to the delivery of every message. In addition, a significant amount of processing is incurred on both the server and the client side to marshal messages. While this marshaling may not necessarily impact the server, it places a heavy burden on the device, especially if the device is on the low end of the performance spectrum. The message import is also limited by the device's nonvolatile storage write performance.
- Subsequent Synchronization – Device-side data changes due to the user interacting with a mobile application. It is important that you understand the unit of change. An operation with an associated tree of objects with a containment (or composite) relationship is considered a unit of change. Changes are wrapped in a message with the appropriate operation type: create or update. (The delete operation requires only the primary key that identifies the root of the tree to be transmitted, rather than the entire object tree). Upon receipt of the message, the server replays the operation against the EIS and returns a replay result message with one or more import messages that reflect the new state of the data. Unlike replication, the unit of change is pushed to the device as soon as the changes are ready, so subsequent synchronization is occurring in the form of many messages. As a result, the synchronization happens over time as a stream of messages.

  The subsequent synchronization phase addresses mobile devices sending CUD requests to Unwired Platform and receiving responses and updates as a result of the operations. In general, the limiting factor for performance is on the EIS as the CUD requests are replayed. You may experience Unwired Server performance issues if requests are spread over a

number of back-end systems. If the EIS response time is large, you may need to allocate additional threads to replay requests concurrently against the EIS.

- Incremental Synchronization – The updates sent to the device as messages due to DCN from the EIS. DCN is the most efficient way for a back-end system to notify Unwired Platform of changes to data mobilized to the device (as opposed to polling for EIS changes).Because messaging uses push synchronization to send out the updates as import messages to the devices, each DCN normally causes updates to device data over the messaging channels, based on a configurable schedule within the server. The main reason for this behavior is the concern for activity storms arising from batched DCNs where many granular changes on the cache cause flurries of messages that might be better consolidated first on the server. Currently, most EIS back-ends are not event-driven and DCNs are batched. Hence, having batched triggers directly driving an event-based model can decrease efficiency without increasing data freshness.

See also

- *Performance Considerations for Replication Payloads* on page 154

Overview of Messaging Performance Recommendations

Messaging tuning recommendations are designed to maximize throughput of messages, as there are unique implications to using messages as the medium of data exchange. The messaging processing limit (throttle) is determined by the number of inbound message queues. Any Unwired Server within a cluster can process messages on inbound queues. The number of inbound queues is equal to the number of parallel messaging package requests that the cluster can concurrently service.

Sybase recommendations touch on components outside and inside the LAN:

- Add multiple server nodes to increase data marshaling. Testing has indicated that one Unwired Server node is capable of sustaining around 70 messages per second to devices. A second Unwired Server node provides an additional 60% increase in delivery capability. This increase is because the second messaging node increases message marshaling and transmission capability for the entire device population.
- Device performance and number of devices deploy. Device processing capacity limits the capacity of each message to 20KB. Due to serialization of data to JSON, a roughly 2X increase in size is observed within our data model, i.e. 4MB worth of data is represented by 9MB of serialized information for packaging into messages. As a result, the number of import messages is in the low 400s. The 2X factor is only an estimate for a moderately complex application; the true cost depends on the number of attributes (and the datatypes) of each MBO in the model. Furthermore, if push messages are to be generated for a device, you should tune the environment after executing download SQL queries against the cache: the cost of the query depends on the complexity of the download logic (number of MBO relationships) in addition to synchronization timestamp comparison. For a large number of devices, this cost can be considerable.
- Approach initial synchronization carefully, and follow a sound rollout policy for new mobile applications. The easiest approach is to spread the rollout over a period a time to

avoid a large number of devices attempting to download large amounts of data. Excessive load not only slows the rollout, but may also impact performance for existing messaging applications. Apart from Unwired Platform capacity and connectivity bandwidth, the actual time required to perform the initial synchronization depends mostly on the capability of the device.

- Allocate additional threads to replay requests concurrently against the EIS. Choosing a value depends upon:
  - The number of concurrent CUD operations that an EIS can handle without causing degradation
  - The number of simultaneously active EIS operations the load is spread across
  - The average response time for various EIS operations
  - Any lock contention points in the mobile model that exist due to shared data partitions
  - The total application landscape (Unwired Platform does not assign threads on a per package basis. All threads are available to service CUD operations for any of the deployed messaging packages.)
- The Unwired Server data change check frequency for each package. The higher the check frequency, the higher the cost, but data freshness may not actually increase. The gating factor is the frequency of updates from the EIS. If the cache is configured to refresh at midnight, after the EIS performs certain batch processing, it is most efficient to configure the scheduler to check for changes sometime after the cache refresh is completed. For an EIS that uses DCN to update the cache, the amount and frequency of changes depends on the business process. Understanding the data flow pattern from EIS to cache is crucial to determine how frequently the Unwired Server is to check for changes to be pushed to the devices.
- Batch multiple DCN updates in a single notification to reduce overhead. However, a batch that is too large may impact device synchronization response due to contention. A storm of DCNs can create significant work for devices. The tradeoff is efficiency versus performance (responsiveness).
- During the packaging of data into messages, serialization consumes a significant amount of memory. Monitor gargage collection activities for Unwired Server. If required, configure JVM heap settings to minimize garbage collection, especially during messaging application deployments and initial synchronizations.

Relay Server and Outbound Enabler Performance Tuning Reference

Use this table to quickly ascertain which Relay Server and Outbound configuration properties you can adjust to tune performance.

Table 20. Recommendations Generally Suitable for 64-bit Production Servers

Component	Function	Default	Production Recommendation
Relay Server shared memory	Settings for shared memory buffer. Remember that shared memory must account for concurrent connections, especially with large payloads.	10MB	2GB
Outbound Enable r deployment	With SQL Anywhere®, each RSOE provides an upload and download channel to the relay server.	1 Outbound Enabler per server	3 Outbound Enablers per synchronization (replication or messaging) port, especially when initial synchronizations are larger than 4MB

Unwired Server Messaging Performance Tuning Reference

Use this table to quickly ascertain which runtime properties you can adjust to tune messaging performance.

The majority of messaging tuning revolves around Unwired Server configuration:

Table 21. Recommendations Generally Suitable for 64-bit Production Servers

Property	Function	Default	Production Recommendation
Number of inbound queue*	The concurrent number of threads that service package(s) requests from devices. If the processor of either the messaging database or the Unwired Server is excessively high, you can throttle the number of requests and limit contention for resources using this setting. Low settings decrease parallelism; high settings may cause undue contention for resources.	5 per cluster	100 per cluster in a 2-node cluster

Property	Function	Default	Production Recommendation
Number of outbound queue*	The number of outbound queues between Unwired Platform and Messaging Services. Messages from these queues are eventually persisted, by the JmsBridge component into a per-device queue belonging to the Messaging Services. Subscription requests can generate a large number of outbound messages and temporarily cause backups in the queues. Since multiple devices can share the same output queue, larger number of queues can reduce delay getting the messages to the intended devices.	25 per cluster	100 per cluster in a 2-node cluster
Check interval for package	Specifies whether the interval system should check for a package to see if there are changes to be pushed to the devices. Align this setting when the cache is being refreshed or updated. If the cache is refreshed every 4 hours, the check interval should also be 4 hours. However, if the cache is only updated via DCN, align the update frequency with the DCN interval accordingly. The check/push generation algorithm is scheduled to be enhanced in future versions.	10 minutes	10 minutes
Number of push processing queues*	Number of queues (threads) that execute the change detection function (download SQL execution) to determine if there are changes to be pushed to the device. The number of queues determines the maximum concurrency for change detection. All packages shared the same set of push processing queues.	25 per cluster	25 per cluster in a 2-node cluster.
JVM minimum heap size	The minimum memory allocated to the differencing and cache management functions of the server.	512MB	2GB

Property	Function	Default	Production Recommendation
JVM maximum heap size	The maximum memory allocated to the differencing and cache management functions of the server.	2GB	6GB

* Cluster-affecting property

Tuning Messaging

Tune your production environment after all components have been deployed and started.

1. Isolate the monitoring database from the cache server.

   See *Isolating and Setting Up Production Monitoring Databases*.
2. Disable all enabled monitoring profiles that currently exist.

   Normally, monitoring in a cluster configuration occurs at two levels — the cluster and the domain. Sybase recommends that you turn off monitoring when assessing performance.

   a) In navigation pane of Sybase Control Center, click **Monitoring**.
   b) In the administration pane, select the profile name and click **Disable**.
   c) Validate that all monitoring is disabled by opening `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\monitoring\MonitoringConfiguration\domainLogging.properties`, and verifying that “status”, “enabled”, and “autoStart” are set to false.

   See *Monitoring Profiles* in the Sybase Control Center online help.
3. Stop all Unwired Servers in your cluster(s).
4. In Sybase Control Center, change the number of outbound and inbound queues to 100 each for each Unwired Server.

   a) In the left navigation pane, click **Servers**>**SeverName**>**Server Configuration**.
   b) In the right administration pane, click the **Messaging** tab.
   c) Change the default values for inbound and outbound queues.
   d) Repeat for all servers in all clusters.
5. Restart all servers in your cluster(s).
6. In Sybase Control Center, deploy and test a messaging package with a representative amount of data for initial subscription.

   For details, see *Deploying and Managing MBO Packages*.

   For example, if the use case dictates that during an initial subscription, the mobile application is to receive 2MB of data, develop the test package to reflect that fact.
7. Start testing by using expected number of devices to perform an initial subscription, and determine if the time to get the initial data set is satisfactory for all devices.

The maximum messaging throughput is 70 messages per second in a wired environment.

- If the calculated throughput for the test is below this number, it is likely that the connection method (as opposed to the server environment) is the limiting factor. In this case, more devices can be supported without any degradation in server performance.
- If the test reaches the maximum throughput, the number of devices performing the initial subscription is the maximum one server can handle. Add another server to the cluster for additional message processing power (up to a 60% increase).

Runtime Maintenance Cleanup

An important part of maintaining system health is periodically cleaning up the runtime environment. Over time, runtime artifacts accumulate and can impact system performance.

Sybase Unwired Platform lets the SUP Administrator and SUP Domain Administrator automate cleanup of accumulated data items at the domain level. Types of data that are eligible for cleanup include: synchronization cache, online cache, client log, error history, and subscriptions.

Set up a schedule for each of these categories, so the clean up processes run when system usage is low. You can also manually purge accumulated data items at any time. You can use the Administration API to automate clean up at the package level.

Sybase also recommends manual cluster-level cleanup for unused devices and users.

Scheduling Domain-Level Cleanup

Periodically clean up accumulated data maintenance items in cache that are no longer needed.

You can automate domain-level cleanup based on a configured schedule for specific cleanup categories.

Running the cleanup options uses system resources, so Sybase recommends that you schedule these tasks when system load is lightest. Optionally you can run the cleanup tasks manually.

1. In the Sybase Control Center left navigation pane, expand the **Domains** tab and select a domain.
2. In the right pane, select the **Scheduled Task** tab.
3. Under Task, select one of the options you want to schedule, and then select **Properties** to set up its automatic schedule:

Option	Description
Subscription Cleanup	Removes subscriptions that are no longer referenced by any active users. • Replication-based synchronization – removes subscriptions not used since the last synchronization. • Message-based synchronization – removes subscriptions if Unwired Server has not received a synchronization message since the given date.
Error History Cleanup	Removes historical data on MBO data refresh and operation replay failures, which result from system or application failures. System failures may include network problems, credential issues, and back-end system failure. Application failures may include invalid values, and non-unique data. **Note:** Only error messages are removed.
Client Log Cleanup	Removes client log records that have already been synchronized to the device, or are no longer associated with active users.
Synchronization Cache Cleanup	Removes logically deleted rows in the cache that are older than the oldest synchronization time on record in the system. Synchronization activity for all clients establish the oldest synchronization time. This cleanup task also removes unused or stale partitions.

4. Select **Enable**. Schedules run until you disable them, or they expire.

See also

- *Maintaining Platform Databases* on page 170

Maintaining Platform Databases

Platform databases need to be rebuilt regularly. Without regular maintenance, the database log can grow to be large (several gigabytes in size), and performance could degrade.

If you experience long shutdown times with data services for the Unwired Platform data tier, you need to unload and reload data the dbunload utility. This rebuilds your database and maintains a healthy data tier performance.

To avoid this issue, Sybase recommends that you perform this action every four or five months.

1. Stop all runtime services (data tier and server nodes), including the cache database, the cluster database, and the messaging database, and synchronization services.

2. Perform a full off-line backup of each, by copying the database and transaction log files to a secure location. See *Backing Up SQL Anywhere Databases*.
3. Rebuild each runtime database with the SQL Anywhere Unload (dbunload) utility.
   The dbunload utility is available in the *UnwiredPlatform_InstallDir*`\Servers\SQLAnywhere11\BIN32` directory.
4. Validate the data before restarting the services.

See also

- *Sample Backup and Recovery Plan* on page 172
- *Scheduling Domain-Level Cleanup* on page 169

Control Transaction Log Size

Control the size of transaction logs to prevent the log files from growing indefinitely.

Use the SQL Anywhere **dbbackup** utility, with the **-xo** flag. The **-xo** flag deletes the current transaction log file, once it has been backed up successfully, and creates a new one. See *SQL Anywhere® Server – Database Administration* for information.

Alternately, use a variant of the SQL Anywhere **BACKUP DATABASE** command. This example performs daily backups automatically from within the database server:

```
CREATE EVENT NightlyBackup
SCHEDULE
START TIME '23:00' EVERY 24 HOURS
HANDLER
BEGIN
   DECLARE dest LONG VARCHAR;
   DECLARE day_name CHAR(20);

   SET day_name = DATENAME( WEEKDAY, CURRENT DATE );
   SET dest = 'd:\\backups\\' || day_name;
           BACKUP DATABASE DIRECTORY dest
   TRANSACTION LOG RENAME;
END;
```

Backup and Recovery

Backup and recovery strategies are part of a larger availability and resiliency strategy you created when designing your network for Unwired Platform environment.

- Availability describes the degree of healthy system operations you can maintain under adverse conditions (for example, if multiple nodes in an Unwired Server cluster become unavailable due to a hardware failure).
- Resiliency describes how quickly healthy system operations return after service degradation.

Backup and recovery strategies can be one of two types: error correction and disaster recovery. The Unwired Platform documentation discusses error correction. For disaster recovery, you

may need to engage with other vendors that provide solutions geared to maintaining the viability of your entire enterprise.

See also

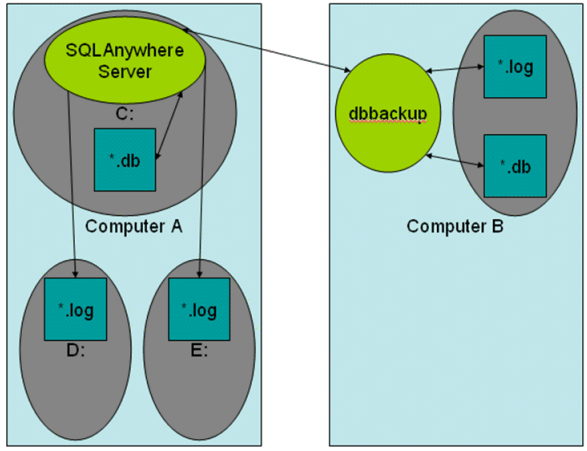
- *Chapter 9, Mobile Data Management Overview* on page 87

Sample Backup and Recovery Plan

Provides a basic backup and recovery plan.

This diagram shows the architecture for a reasonably reliable backup and recovery strategy. Only Sybase Unwired Platform components related to database recovery are included.

Figure 1: Sample backup and recovery plan



Shown in the diagram:

- Computer A is where the SQL Anywhere database server is installed and runs under Sybase Unwired Platform. There are three physical disks on this computer:
  - The C: drive has the SQL Anywhere server and the database files (`default.db`, `clusterdb.db`, and `monitor.db`), which hold critical data that Sybase Unwired Platform requires to function.
  - The D: and E: drives hold identical (that is, mirrored) versions of the transaction log files (`default.log`, and `clusterdb.log`). Using SQL Anywhere terminology, the D: drive holds the regular transaction log, and the E: drive holds the mirrored transaction log.
- Computer B is for long term backup, and requires only one drive (or backup tapes). Run the **dbbackup** utility from this computer periodically to obtain a full backup of the `*.db` and `*.log` files from Computer A.

See also

- *Using a Different Database Log Path* on page 22
- *Maintaining Platform Databases* on page 170

Failure and Recovery Scenarios

Describes several failure scenarios (using the Sample Backup and Recovery Plan setup), how recovery works, and implications for Sybase Unwired Platform operation.

*Disk C has an unrecoverable error. The *.db files have been lost.*
Recovery:

1. Make sure you have current backups of `*.db` and `*.log` files; in this scenario they are on Computer B.
2. Install a replacement disk, and use the Sybase Unwired Platform runtime installer to reinstall the data tier. This restores the embedded SQL Anywhere software, used as the cache database for synchronization.
3. Restore the database `*.db` files, by copying the last backup version of the files from Computer B.
4. Restore the transaction `*.log` files.
   - If you are restoring a single log, you can use the -a option, using a command similar to:

     ```
     dbeng<sqlAnywhere version> <path>default.db -a
     <restored_path>default.log
     ```

     Note: See *Supported Hardware and Software* for supported SQL Anywhere versions embedded with Sybase Unwired Platform.
   - If you are restoring multiple transaction logs, they must be started in the correct order, otherwise, the database does not start. You can use the -ad or -ar option in your command to automatically apply multiple transaction logs in the correct order, using a command similar to:

     ```
     dbeng<sqlAnywhere version> <path>default.db -ad
     <restored_path>default.log
     ```

   In either case, see the SQL Anywhere documentation for information about composing commands to restore transaction logs. O.
   - *Recovering a database with multiple transaction logs* section – *http://dcx-internal.sybase.com/index.html#1201/en/dbadmin/backup-s-4334998.html*
   - *Recovering from Media Failure* section – *http://dcx-internal.sybase.com/index.html#1201/en/dbadmin/da-backup-s-5120338.html*
5. Start Unwired Server, which restarts SQL Anywhere and detects that the `*.db` files are not up-to-date with the checkpoints in the `*.log` files on drives D: and E: (which are unaffected by the C: drive failure). The server automatically replays transactions recorded

in the transaction log to bring the database back to the state of all committed transactions at the time of the C: drive failure.

Sybase Unwired Platform can then start up and run normally. Sybase Unwired Platform mobile device clients are not affected except by the inability to synchronize between the time of the failure, and the time at which the recovery process has completed.

*Disk D: or E: failure. One of the *.log files has been lost.*
Recovery: Install a replacement disk and restore from backups.

Once the disk has been restored, copy the `*.log` files from the drive that did not fail to the one that failed. Restart the failed drive.

Complete failure of Computer A, and disks lost.
Recovery: See *Restoration of the Runtime Database* for information. This should be a very infrequent event.

In this scenario, the database has lost all transactions since the previous backup to Computer B. Any Sybase Unwired Platform mobile device clients that synchronized between the time of the previous backup and the time of the failure cannot now sync. Clients must delete their remote device client database and start fresh. Any pending operations on these clients are lost. Clients that have not synchronized since the previous backup are unaffected.

Backing Up the File System

Regularly perform backups of the Sybase Unwired Platform files and directories.

Avoid backing up individual artifacts. Instead, Sybase recommends that you perform regular backups of entire directories as part of your disk backup schedule.

Note: At the same time the folder and disk backup is performed, update the Windows registry.

1. Plan the frequency of the file system backups to coincide with any changes made to the system, including:
   - Metadata changes (such as deployment of new mobile business object packages to the server)
   - Configuration changes (such as new enterprise information system connection)
2. Back up application artifacts, by using Sybase Control Center to export packages. This preserves each deployed package in an alternate location. See *Exporting Package Content* in Sybase Control Center online help.
3. Back up the contents in these Unwired Server directories:
   - `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin`
   - `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\config`
   - `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\logs`

- `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository`

4. Back up the contents of your system servers:
   - `<UnwiredPlatform_InstallDir>\Servers\Advantage910`
   - `<UnwiredPlatform_InstallDir>\Servers\MessagingServer`
   - `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11`
5. Back up the contents of these Sybase Control Center directories:
   - `<UnwiredPlatform_InstallDir>\SCC-X_X\conf`
   - `<UnwiredPlatform_InstallDir>\SCC-X_X\services\repository\repository.db`
   - `<UnwiredPlatform_InstallDir>\SCC-X_X\log`

Next

To maintain a consistent backup state, Sybase recommends you back up the data cache database at these times as well. See the next task (Backing Up System Data) in this sequence.

Backing Up System Data

For platform data, back up Unwired Server runtime databases and Sybase Control Center (SCC) repositories using the process described for SQL Anywhere databases. Messaging database requires its own process.

Table 22. Runtime Database Default File Locations

Runtime Databases	Default File Locations

Unwired Server	For a Developer Edition installation, database files and transaction logs are installed:
	`<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\data`
	• Cache database:
	  • Database file: `default.db`
	  • Transaction log: `default.log`
	• Cluster database:
	  • Database file: `clusterdb.db`
	  • Transaction log: `clusterdb.log`
	• Monitor database:
	  • Database file: `monitordb.db`
	  • Transaction log: `monitordb.log`
	For a separate Data-tier node, these database files and transaction logs are installed:
	• With a Microsoft cluster, in the Microsoft cluster folder you created: *<Microsoft_cluster_folder>*`\CDB`
	• With no Microsoft cluster: *<UnwiredPlatform_InstallDir>*`\UnwiredPlatform\Data\CDB`

Sybase Control Center	Database files and transaction logs on each Unwired Server node: `<UnwiredPlatform_InstallDir>\SCC-`*X_X*`\services\Repository` • Database file: `scc_repository.db` • Transaction log: `scc_repository.log`

Note: When you make a backup, decide where to store the backup files: on the Unwired Server host or on some other computer or third-party hardware/software package used for backup purposes.

Backing Up SQL Anywhere Databases

Back up platform data warehoused in SQL databases using SQL Anywhere utilities.

Prerequisites

On backup hosts, verify that SQL Anywhere software is installed, and the PATH is set. If you are running Sybase Unwired Platform as a cluster, SQL Anywhere is installed under the Sybase Unwired Platform installation directory. Otherwise, install another copy of Sybase Unwired Platform on backup machines. Otherwise, SQL Anywhere components may not work as expected.

Task

By default, Unwired Platform uses SQL Anywhere for the cache and cluster databases. `clusterdb` is present if Unwired Server has been configured to run as a cluster. Use various SQL Anywhere utilities to help you with these tasks:

- **dbvalid** – validates the integrity of the database by checking the index keys on some or all of the tables in a database, as well as on the database file itself.
- **dblocate** – locates any SQL Anywhere database servers (cache or cluster) running over TCP/IP on the immediate network, and prints a list of the database servers and their addresses. This list includes alternate server names. Depending on your network, it may take several seconds for **dblocate** to print its results.
- **dbbackup** – makes a backup copy of all files for a single database from a remote backup location. A simple database consists of two files: the main database file and the transaction log. If you mirrored the transaction log you need not back up this file. All backup file names are identical to database file names. The image backup contains a separate file for each backed-up file.
- **dblog** – mirrors the transaction log used for these databases.

Backing Up Database Files

Regularly perform a remote backup of data so it can be recovered. Before you back up system data, ensure the data is valid.

Note: Depending on the design of your production environment, the **dbvalid** and **dbbackup** commands may not work remotely.

The database server is named `<clustername>_primary`.

---

1. Validate the databases:
   a) Ensure there are no active connections to the server.
   b) Validate the cache database:

      ```
      dbvalid.exe -c "DBF=default.db;UID=dba;PWD=SQL"
      ```

   c) Validate the cluster database:

```
dbvalid.exe -c "DBF=clusterdb.db;UID=dba;PWD=SQL"
```

d) Validate the monitoring database:

```
dbvalid.exe -c "DBF=monitor.db;UID=dba;PWD=SQL"
```

2. On the backup machine, verify that SQL Anywhere software is installed, and the PATH is set.
3. Back up the databases to archive the system data:
   - For cache databases, run:

     ```
     dbbackup -c
     "ENG=<clusterName>_primary;DBN=default;UID=dba;PWD=SQL"
     \SQLAnybackup
     ```

   - For cluster databases, run:

     ```
     dbbackup -c
     "ENG=<clusterName>_primary;DBN=clusterdb;UID=dba;PWD=SQL"
     \SQLAnybackup
     ```

   - For monitor database, run:

     ```
     dbbackup -c
     "ENG=<clusterName>_primary;DBN=monitor;UID=dba;PWD=SQL"
     \SQLAnybackup
     ```

   This creates `default.db`, `default.log`, `clusterdb.db`, and `clusterdb.log`, and `monitor.db`, `monitor.log` in the `\SQLAnybackup` directory on the backup computer.
4. As a precaution, validate the backups are suitable for recovery:
   a) On the backup computer, create a temporary working directory (such as `\tmp`).
   b) Under the temporary directory, create an identical directory structure for the two log locations. You may need to use the **subst** command to map local directories to drive letters used on the runtime computers to the backup location.
   c) Copy `*.log` to these locations.
   d) Run **dbvalid** on the `\tmp` copy of the `.db` file.

      WARNING: Do not run **dbvalid** on the backup copy itself (in the `\SQLAnybackup` directory of this example). The command runs, but corrupts your `.db` file so it cannot be used in recovery.
      - If validation succeeds, the backup in `\SQLAnybackup` can be used for recovery; delete the files in the `\tmp` and log directories.
      - If validation fails, the backup is not usable for recovery; try again.

Next

With the archive of the database complete, you can optionally back up the archive to a tape drive.

Backing up Data and Mirroring Transaction Logs

Mirror logs to make identical copies elsewhere on your network using SQL Anywhere utilities. Mirrored logs ensure runtime data is available and recoverable in the event of failure.

Use dblog command utility to set up log mirroring. Sybase recommends an extension of `*.mlg` for the mirrored log so the mirrored log can quickly be identified.

Mirror the transaction log, so that if database file becomes unrecoverable, you can use record of changes since the last backup in the transaction log to rebuild the database.

From the `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32` directory run the dblog command with the -m option: For example, to mirror the original log file on drive D:\ to drive E:\ use:

```
dblog -t D:\logs\default.log -m E:\logs\default_mirror.mlg
default.db
```

Backing Up Messaging Data

Use the **adsbackup** utility to back up messaging data. You can then use MORecover to recover the data to an existing database.

The backup target can be any folder. A collection of files constituting the backed-up data is created in a folder you specify.

Note: Backup files only include raw data in the original tables, without the index data. Therefore, they cannot be used directly as a database and must be recovered with MORecover before they can be used.

You can schedule this command to run regularly when your system is lightly loaded, since the backup operation runs simultaneously with the messaging server's normal processing.

1. Change to `<UnwiredPlatform_InstallDir>\Servers\AdvantageXXX\Server`.
2. Run the **adsbackup** utility to create a MORecover snapshot; for example:

   ```
   adsbackup.exe
   iAPPLICATIONS,CFG_IDS,CFG_PROP_VALUES,CFG_SUBFOLDER_PROP_V
   ALUES,CFG_TEMPLATES,DEVICES,USER_DEVICE,USERS "C:\Sybase
   \UnwiredPlatform-X_X\Servers\MessagingServer\Data\OBR
   \OBR.add" <MYBackupTarget>
   ```

See also

- *Advantage Database Server® Backup (adsbackup) Utility* on page 246

Restoration of the Installation File System

Restore the Sybase Unwired Platform installation file system from a backup.

To perform a normal restoration, use the file or disk backup utilities used to perform the backup. Sybase recommends that you save the current installation directory before you restore from backup.

Note: You may also need to restore the Windows registry from the backup done at the same time.

Restoration of the Runtime Database

Restore the Sybase Unwired Platform runtime databases and transaction logs from a backup.

As discussed in *Failure and Recovery Scenarios*, if only one of C:, D:, or E: drives fail, recovery should be automatic once you have completed the appropriate tasks.

These steps are required in case of complete failure of all three drives:

1. Restore the computer's C:, D:, and E: drives from backup.
2. Delete the `*.db`, `*.log` files from their normal places after you have restored the file system.
3. Copy the `*.db` and `*.log` files from Computer B's backup directory to the normal locations on Computer A.

   Note: Copy the *.log files twice—once to the normal transaction logs directory, and once to the mirrored transaction logs directory.
4. Restart Sybase Unwired Platform.
5. If there have been package deployments or other cluster-affecting operations since the last database backups, the file-system data corresponding to the packages may be out-of-sync with the database contents related to these packages. If this has occurred, the Sybase Unwired Platform servers cannot fully start. Check server bootstrap log and the <hostname>-server.log file. These logs include server mismatch messages that prevents server from starting normally. To recover from this:
   a. Choose one of the Unwired Servers.
   b. Shut down that server.
   c. Edit the `sup.properties`, and change the `cluster.version` property value to match that of the current cluster as reported in the logs. This file is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer`.
   d. Run `updateProps -r` from the same directory to apply the change into the `clusterdb`.
   e. Restart that Unwired Server.

   Note: This server should be able to take over as the Sybase Unwired Platform primary. Sybase recommends you redeploy all your packages (using the **UPDATE** option) to make

sure all the packages you expect to be available really are. All of your Sybase Unwired Platform clients should delete their UltraLite databases, and perform full synchronizations.

Restoration of the Messaging Data

Restore the Sybase Unwired Platform messaging data tables from a backup by running the MORecover utility.

You can find this utility in the `<UnwiredPlatform_InstallDir>\Servers\Messaging\Server\Bin` folder.

Before running MORecover to restore your database, convert the data files created by **adsbackup** back to a standard database format, by running **adsbackup** with the -r option. You can then use MORecover to restore the backed-up data.

For example, if you sent your backed-up data output to `c:\bak`, but restored the data to `c:\bak\restore`, the command line syntax for **adsbackup** is:

```
adsbackup.exe -r "c:\bak\obr.add" "c:\bak\restore\obr.add"
```

When you run the MORecover utility, use the restore location. For example:

```
MORecover "c:\bak\restore\obr.add"
```

See also

- *Unwired Server Database File Recovery (MORecover) Utility* on page 248

Sybase Unwired Platform Licenses

Sybase Unwired Platform is available in three editions.

Edition	Summary
Personal Development Server	• Allows use in development systems and testing systems only; not for use in production systems. • Requires all Unwired Platform server components to be installed on the same, single-user host with Sybase Mobile SDK. • Allows a maximum of five mobile client devices.
Enterprise Development Server	• Allows use in development systems and testing systems only; not for use in production systems. • Allows each installable component to be located on a separate host. • Allows clustered systems. • Allows a maximum of 20 mobile client devices.

Edition	Summary
Enterprise Server	• License type determines allowed use (production only, or development and testing only). • Allows each installable component to be located on a separate host. • Allows clustered systems. • Requires separate license for mobile client devices (production). • Unlimited mobile client devices (development and testing).

Cluster License Coordination

In a cluster, each server deployed to the environment must be licensed. Multiple servers cannot share a single license. However, all server nodes in the cluster can share device connection licenses.

In a clustered environment, you must use a license server so it can coordinate licensing requirements among all installed components:

- Server validation – each time a server starts, it connects and registers with the license server to check if there is a valid license for it. If there is a free license available, the server checks out the license and continues with the start-up process. If the number of licensed servers cannot be retrieved from the license file, or the license server confirms that a server is not licensed, Unwired Server stops.
- Device connection validation – because available device licenses are shared among all servers in the cluster, all connections to all servers must be accounted for. The cluster name enumerates each device connection made across clustered servers. Every server then checks out all device licenses when the servers start.

License Validation

Attributes in the license file control the base number of devices that can be registered, the number of servers (typically for clustered production environments) you install, and expiry dates for both devices and servers. The mechanism that counts device licenses varies, depending on your model.

There are two licensing models you can use with Unwired Platform:

- Unserved (local) license – uses a local license file for each Unwired Platform installation.
- Served (SySAM license server) – uses a SySAM license server to support multiple Unwired Platform installations.

For both models, Unwired Server always tracks available licenses and expiry dates, and writes license errors to the Unwired Server log. Administrators can always check these limits and take appropriate action when that limit is reached.

Unserved Model

In an unserved license model, licenses are validated at several intervals:

- At start-up – if Unwired Server cannot retrieve the number of licensed servers from the license file, or if the server is not licensed, Unwired Server stops.
- At device connection – when the device user tries to connect to Unwired Server, Unwired Server checks the device ID against the data tier. If the device falls within the device license limit, the device connection continues and operations proceed normally for both replication and messaging applications. If the device falls outside the limit, Unwired Server throws a license check exception to the client. See *Device License Limits* in *System Administration*.
- Upon license expiry – if the date in the license file matches the current date, the license expires; Unwired Server generates a license expired error. The error text varies, depending on whether the server or the client connection licenses have expired. If a server license is expired, Unwired Servers also stop.

Served Model

In a served license model, licenses are validated at these intervals:

- At start-up – if Unwired Server cannot retrieve the number of licensed servers from the license file, or if the server is not licensed, Unwired Server stops.
- With each synchronization – the procedure varies slightly depending on the synchronization model used on the client:
  - For replication synchronization – after the device user is authenticated, Unwired Server uses the device ID to check the license into the data tier. If the device falls within the device license limit, synchronization proceeds. If the device falls outside the limit, Unwired Server throws a license check exception to the client.
    Administrators must monitor licenses carefully; there may be many devices connected to the server, but fewer licenses being used. See *Device License Limits* in *System Administration Guide*.
  - For messaging synchronization – when the device user tries to connect, Unwired Server checks the device ID against the data tier. If the device is registered, and the total number of devices registered falls within the device license limit, the message is processed normally. If the device is not registered, or the total number of devices registered falls outside the limit, Unwired Server throws a license check exception to the client.
- Upon license expiry – if the license expires, Unwired Server generates a license expired error. The error varies, depending on whether the server or the client connection licenses have expired. When a server license expires, Unwired Servers also stop.

Device License Limits

Licenses limit how many components you can install on a network, and how many users can connect to your servers.

If you notice messages similar to these errors in the Unwired Server log, then connection requests from registered users have exceeded the licensed limit:

```
2009-12-28 18:01:59.872 INFO    MMS          Thread-19
[com.sybase.sup.server.lm.LicenseUtil] The number of registered
```

```
devices has reached the maximum limit for your license.
2009-12-28 18:01:59.965 INFO    MMS          Thread-19
[com.sybase.djc.mobilink.LoginHandler] The number of registered
users has reached the maximum limit for your license.
2009-12-28 18:02:00.168 ERROR   Mobilink     Thread-19
[com.sybase.ml.sup.Logger] [-10052] authenticate_parameters scripts
return 4000
```

For example, a trial license limits you to only five device users. If a sixth user tries to connect, the error is logged accordingly.

In cases where the number of users in your environment exceeds that of your license, you can:

- Upgrade your license and manually change the license in your environment.
- Control the number of device user connections at a given moment in SCC. For example, you can view the total number of users in the User Statistics tab of the Monitor node.
  If the number of device users is too high for your license, you can manually delete unused devices to make room for new users:
  - For workflow and Online Data Proxy client devices, delete the Application Connections that are no longer in use.
  - For Native replication applications, delete Package Users on the respective Package.
  - For native messaging client applications, delete the Application Connections associated with the native messaging clients not in use.

  See the *Deleting Application Connections* and *Deleting a Package* topics in *Sybase Control Center* online help.

Checking System Licensing Information

Review licensing information to monitor available and used device licenses, license expiry dates, and other license details. This information allows administrators to manage license use and determine whether old or unused device licenses should be transferred to new devices.

1. In the left navigation pane, select the top-level tree node.
2. In the right administration pane, select the **General** tab, and click **Licensing**.
3. Review the following licensing information:
   - Server license type – the type of license currently used by Unwired Platform. For more information on license types, see *Sybase Unwired Platform Licenses*.
   - Production edition – the edition of the software you have installed.
   - Server license expiry date – the date and time at which the server license expires. When a server license expires, Unwired Server generates a license expired error and Unwired Server is stopped.
   - Overdraft mode – allows you to generate additional licenses in excess of the quantity of licenses you actually purchased. This enables you to exceed your purchased quantity of licenses in a peak usage period without impacting your operation. This mode is

either enabled or disabled, as specified by the terms of the agreement presented when you obtain such a license.

- Total device license count – the total number of device licenses available with your license. This count limits how many devices can connect to your servers.
- Used device license count – the total number of unique devices associated with the users currently registered with the server. If all of your available device licenses are in use, you can either upgrade your license or manually delete unused devices to make room for new users:
  - For workflow and Online Data Proxy client devices, delete the Application Connections that are no longer in use.
  - For native replication-based applications, delete Package Users on the respective Package.
  - For native messaging-based applications, delete the Application Connections associated with the clients not in use.
- Device license expiry date – the date and time at which the device license expires. When a device license expires, Unwired Server generates a license expired error and connection requests from registered devices are unsuccessful.
- Used mobile user license count – the number of mobile user licenses currently in use. A mobile user is a distinct user identity—username and associated security configuration—that is registered in the server. As such, the used mobile user license count represents the total distinct user identities registered on the server. One mobile user may access:
  - Multiple applications and different versions of the same application.
  - The same or different versions of an application from multiple devices.
- Used application user license count – the number of all registered application users of all applications. This value represents the cumulative total of the distinct user identities registered for each application. The same user identity using:
  - Multiple versions of the same application counts as one application user.
  - Two different applications count as two application users.

4. Click **Close**.

Note: Unwired Platform licensing is configured during installation. However, if necessary, license details can be changed at a later time. See *Manually Updating and Upgrading License Files*.

Manually Updating and Upgrading Licenses

You must upgrade Unwired Platform and Afaria licenses separately. For Unwired Platform upgrades, the procedure varies depending on whether you are using a served or unserved model.

To update or upgrade your license, ensure you have the updated or upgraded license file and save it in the appropriate location for your served or unserved license model. Then you need to run `license.bat` and restart Unwired Servers for license change to take affect.

1. *Updating and Upgrading Unwired Platform Licenses*

   The <UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin\license.bat script allows you to upgrade Unwired Platform licenses without having to re-run the installer. You can use this script in both served and unserved license models.

2. *Upgrading Afaria Licenses*

   If you have a version of Afaria installed with an earlier version of Unwired Platform, and need to upgrade to a standalone version of Afaria with more license options enabled, you need to change the serial number before they are enabled.

Updating and Upgrading Unwired Platform Licenses

The `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin\license.bat` script allows you to upgrade Unwired Platform licenses without having to re-run the installer. You can use this script in both served and unserved license models.

Prerequisites

For unserved models, ensure that you have already copied the new license file to the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\licenses` folder.

Task

1. Use the license to locate information that the `license.bat` script requires. See *Locating Information in a License File*.

   When you set up your license model, Sybase license fulfillment generates a new Unwired Platform license that includes a new serial number. For subsequent upgrades, use the same serial number to incrementally update licenses.

2. From the command prompt, change to the `bin` folder for Unwired Server.
   For example, `cd C:Sybase\UnwiredPlatform\Servers\UnwiredServer\bin\`.

3. Run `license.bat`:

   `license.bat PE LT <licenseNumber>`,

   where `PE` is the Production Edition, `LT` is the License Type, and `<licenseNumber>` are the number of licensed devices supported.

See also

- *Locating Information in a License File* on page 187
- *License Upgrade (license) Utility* on page 238
- *Upgrading Afaria Licenses* on page 189

Locating Information in a License File

After you download a license file, you may need to get some information from it to complete your installation.

1. Use a text editor to open your license file.
2. Locate the uncommented line that begins with the string for your Unwired Platform edition:
   - Enterprise Edition – `INCREMENT SUP_ENTSRVR`
   - Enterprise Developer Edition – `INCREMENT SUP_ENTDEV`
   - Personal Developer Edition – `INCREMENT SUP_DEVELOPER`

   For example:
   - Enterprise Edition would be similar to this.

     ```
     ...
     INCREMENT SUP_ENTSRVR SYBASE 2011.11150 permanent uncounted \
        VENDOR_STRING=PE=EE;LT=CP HOSTID=000c29d300bd
     PLATFORMS="i86_n \
     ...
     ```
   - Enterprise Developer Edition would be similar to this.

     ```
     ...
     INCREMENT SUP_ENTDEV SYBASE 2011.11150 permanent uncounted \
        VENDOR_STRING=PE=EE;LT=CP HOSTID=000c29d300bd
     PLATFORMS="i86_n \
     ...
     ```
   - Personal Developer Edition would be similar to this.

     ```
     ...
     INCREMENT SUP_DEVELOPER SYBASE 2011.11150 permanent uncounted \
        VENDOR_STRING=PE=EE;LT=CP HOSTID=000c29d300bd
     PLATFORMS="i86_n \
     ...
     ```

   The rest of the examples in this section show the beginning of this line as it would appear for Enterprise Edition. The details illustrated apply equally to all editions.
3. Determine whether the server license is served or unserved.

   If the line you located in step 2 ends with "uncounted" it is an unserved license. For example:

   ```
   ...
   INCREMENT SUP_ENTSRVR SYBASE 2011.11150 permanent uncounted \
      VENDOR_STRING=PE=EE;LT=CP HOSTID=000c29d300bd PLATFORMS="i86_n
   \
   ...
   ```

   If that line ends with a number immediately following a date, it is a served license. For example:

   ```
   ...
   INCREMENT SUP_ENTSRVR SYBASE 2011.11150 permanent 10 \
   ```

```
   VENDOR_STRING=PE=EE;LT=CP HOSTID=000c29d300bd PLATFORMS="i86_n
\
...
```

4. Determine the product edition and license type for the license.

   For both served and unserved licenses, note the value of PE (product edition) and LT (license type) in the line following the line you located in step 2. For example:

   ```
   ...
   INCREMENT SUP_ENTSRVR SYBASE 2011.11150 permanent uncounted \
     VENDOR_STRING=PE=EE;LT=CP HOSTID=000c29d300bd PLATFORMS="i86_n
   \
   ...
   ```

   The PE value is the license product edition value; "EE" in the example above.

   The LT value is the license type value; "CP" in the example above.

5. If you are installing Enterprise Edition, determine the number of client licenses.

   If your license type is Development and Test (DT), you can change this number later.

   a) Locate the uncommented line, beginning with INCREMENT SUP_ENTCLIENT.

      For example:

      ```
      INCREMENT SUP_ENTCLIENT SYBASE 2011.11150 permanent uncounted \
               VENDOR_STRING=PE=EE;LT=ST HOSTID=000c29d300bd
      PLATFORMS="i86_n \
      ...
      ```

   b) Determine whether the client licenses are served or unserved.

      If the line beginning with INCREMENT SUP_ENTCLIENT ends with "uncounted" the client licenses are unserved. For example:

      ```
      INCREMENT SUP_ENTCLIENT SYBASE 2011.11150 permanent uncounted \
         VENDOR_STRING=PE=EE;LT=ST HOSTID=000c29d300bd
      PLATFORMS="i86_n \
         x64_n" ISSUER="CO=Sybase,
      Inc.;V=2.0;AS=A;MP=3120;CP=100;EGO=" \
      ...
      ```

      If that line ends with a number immediately after a date, the client licenses are served. For example:

      ```
      INCREMENT SUP_ENTCLIENT SYBASE 2011.11150 permanent 100 \
          VENDOR_STRING=PE=EE;LT=ST HOSTID=000c29d300bd
      PLATFORMS="i86_n \
      ...
      ```

   c) Determine the number of client licenses.

      For unserved client licenses, the number of client licenses is the value of CP two lines below the line beginning with INCREMENT SUP_ENTCLIENT. For example:

      ```
      INCREMENT SUP_ENTCLIENT SYBASE 2011.11150 permanent uncounted \
          VENDOR_STRING=PE=EE;LT=ST HOSTID=000c29d300bd
      PLATFORMS="i86_n \
          x64_n" ISSUER="CO=Sybase,
      ```

```
Inc.;V=2.0;AS=A;MP=3120;CP=100;EGO=" \
...
```

For served client licenses, the number of client licenses is the value at the end of the line beginning with INCREMENT SUP_ENTCLIENT. For example:

```
INCREMENT SUP_ENTCLIENT SYBASE 2011.11150 permanent 100 \
    VENDOR_STRING=PE=EE;LT=ST HOSTID=000c29d300bd
PLATFORMS="i86_n \
...
```

See also

- *Updating and Upgrading Unwired Platform Licenses* on page 186
- *License Upgrade (license) Utility* on page 238

Upgrading Afaria Licenses

If you have a version of Afaria installed with an earlier version of Unwired Platform, and need to upgrade to a standalone version of Afaria with more license options enabled, you need to change the serial number before they are enabled.

The server configuration area of the Afaria Administrator lets you define parameters for the Afaria Server, including new license data.

1. Obtain a new serial number.

   Sybase license fullfilment generates an Afaria license string that includes a new serial number. For subsequent option upgrades, use the same serial number to incrementally update licenses.

2. Define new software licenses in Afaria Administrator.

   The License Compliance page appears empty until you define licensing details. Once you have defined these software licenses, the page shows data for Client category, Manufacturer, Application, Size, Version, # (number) Purchased, Effective and Expiration dates, and any Notes you add. Initially, licenses are sorted by Client category, Manufacturer, then Application. See *Afaria Reference | Platform > Server Configuration > License Compliance*.

See also

- *Updating and Upgrading Unwired Platform Licenses* on page 186

CHAPTER 11 Unwired Server Management API

Sybase Unwired Platform includes a Java API that opens the administration and configuration of Sybase Unwired Platform to Java client applications you create. By building a custom client with the administration client API, you can build custom application to support Sybase Unwired Platform administration features and functionality within an existing IT management infrastructure.

The client you create connects to Unwired Server via Sybase Control Center embedded management server and Sybase Control Center. For details, see *Developer Guide for Unwired Server Management API*.

The administration client API includes these administration interfaces:

Interface	Includes methods that
`SUPServer`	Command and control operations for an Unwired Server instance, for example start, stop, and ping.
`SUPCluster`	Manage cluster security, monitoring configuration and domain creation for a cluster instance, and so on.
`SUPDomain`	Manage domains, deploy packages to a domain, set security configurations for a domain, and so on.
`SUPPackage`	Configure packages by setting up subscriptions, configuring cache groups, configuring endpoint properties, and so on.
`SUPMobileBusinessObject`	View mobile business object properties, operations, errors, endpoints, and so on.
`SUPOperations`	View operation properties, errors, endpoints, and so on.
`SUPApplication`	Manage applications, application connections, and application connection templates
`SUPMonitor`	Perform monitoring functions like viewing histories, summaries, details, and performance data for various platform components, and export data as required.
`SUPServerLog`	View, filter, delete and refresh logs, configure appenders, and so on, for Unwired Server and its embedded services like replication and messaging synchronization.
`SUPDomainLog`	Configure domain log settings and view, filter, delete domain logs entries, and so on.

Interface	Includes methods that
`SUPServerConfiguration`	Configure an Unwired Server instance, as well as its listeners. All methods of this interface, except the apple push notification-related properties are metadata-based.
`SUPSecurityConfiguration`	Create, manage, and configure a security configuration with at least one authentication provider. You can add other providers (authentication, authorization, attribution, and audit) as required.
`SUPMobileWorkflow`	Manage and configure deployed mobile workflow packages.

Javadocs

The administration client API installation includes Javadocs. Use the Sybase Javadocs for your complete API reference.

As you review the contents of this document, ensure you review the reference details documented in the Javadoc delivered with this API. By default, Javadocs are installed to `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\AdminClientAPI\com.sybase.sup.adminapi\docs\api\index.html`.

The top left navigation pane lists all packages installed with Unwired Platform. The applicable documentation is available with `com.sybase.sup.admin.client` package. Click this link and navigate through the Javadoc as required.

CHAPTER 12 SNMP Notifications

(Not applicable to Online Data Proxy)You can set up Sybase Control Center *X.X* to include a Simple Network Message Protocol (SNMP) plug-in that sends notifications to the configured target when the state of an Unwired Server changes (that is, from running to stopped, or from stopped to running).

SNMP is the standard protocol for managing networks and exchanging messages. If the SNMP plug-in is set up for a Sybase Control Center *X.X*, the plug-in creates notifications in response to predetermined status change events that are detected and signaled by the Unwired Server code. When the plug-in generates a notification, a single copy of the notification is transmitted to each target. The SNMP notification target must be the host name and port of a network monitoring station (NMS) that has the ability to process SNMP notifications; Unwired Platform does not include this functionality. Targets and other notification configuration information are read when the Sybase Control Center *X.X* is initialized; therefore, you must stop and restart the agent when enabling SNMP.

Setting Up SNMP Notifications

Setting up SNMP notifications requires you to modify the configuration for Sybase Control Center *X.X* and correctly configure an existing SNMP network monitoring station (NMS). Always test the implementation to validate its setup.

1. *Enabling SNMP Notifications for Unwired Platform*

   Enable the Unwired Platform SNMP plug-in for Sybase Control Center and set up a notification target to receive messages when the status of a local Unwired Server changes.
2. *Handling Transmitted SNMP Notifications*

   Configure an SNMP notification handling tool to process Unwired Platform SNMP notifications.
3. *Testing Notifications with SNMP Queries*

   Test the configuration of the Unwired Platform SNMP plug-in for Sybase Control Center by using a management information base (MIB) browser tool to execute an SNMP query.

Enabling SNMP Notifications for Unwired Platform

Enable the Unwired Platform SNMP plug-in for Sybase Control Center and set up a notification target to receive messages when the status of a local Unwired Server changes.

Prerequisites

Before modifying the SNMP `agent-plugin.xml` and `service-config.xml` files, stop the Sybase Control Center *X.X* service.

Task

Note: The SNMP plug-in for Sybase Control Center detects the status of only the Unwired Server running on the same host computer as your instance of Sybase Control Center.

1. Stop Sybase Unified Agent.
2. Enable the SNMP plug-in to run when Sybase Control Center starts:
   a) Open `<UnwiredPlatform_InstallDir>\<SCC-X_X>\plugins\com.sybase.supsnmpplugin_X.X.X\agent-plugin.xml`.
   b) Set `register-on-startup="true"`.
   c) (Optional) Modify the value of the `sup.server.ping.schedule.interval` property to specify how often (in seconds) the SNMP plug-in pings Unwired Server to detect the server status. The default is 100.
3. (Optional) Change the SNMP notification target to a custom destination:
   a) Open `<UnwiredPlatform_InstallDir>\<SCC-X_X>\services\Snmp\service-config.xml`.
   b) Modify the `snmp.notification.targets` property as follows:

      ```
      set-property property="snmp.notification.targets"
      value=<hostname or IP>/<port number>
      ```

      For example, `set-property property="snmp.notification.targets" value="127.0.0.1/162,10.42.33.136/49152"`. *<hostname or IP>* indicates the server where the network monitoring station (NMS) is located. *<port number>* specifies the SNMP notification port of the NMS. The default SNMP notification port is 162. As indicated in the example, you can set multiple SNMP notification targets.
4. Restart the agent.

Handling Transmitted SNMP Notifications

Configure an SNMP notification handling tool to process Unwired Platform SNMP notifications.

Prerequisites

Install an SNMP notification handling tool, such as HP OpenView.

Task

1. On the SNMP notification target host computer, launch an SNMP notification handling tool.
2. Using the third-party documentation, configure the SNMP notification handling tool to process Unwired Platform SNMP notifications. Configure the notification handler to listen for notifications on the port specified in the "snmp.notification.targets" property of the `<UnwiredPlatformInstall>\<SCC-XX>\services\Snmp\service-config.xml` file.

Testing Notifications with SNMP Queries

Test the configuration of the Unwired Platform SNMP plug-in for Sybase Control Center by using a management information base (MIB) browser tool to execute an SNMP query.

Prerequisites

Install a third-party MIB browser tool.

Task

1. Load `<UnwiredPlatformInstall>\<SCC-XX>\plugins\com.sybase.supsnmpplugin_1.5.2\SYBASE-SUP-MIB.txt` as a module into your MIB browser.
2. Configure the MIB browser settings to use an SNMPv3 information module with the parameters specified in the `<UnwiredPlatformInstall>\<SCC-XX>\services\Snmp\service-config.xml` file:

Property name	Description	Default value
snmp.transport.mappings	SNMP agent host/port	UDP (0.0.0.0/1498) The host "0.0.0.0" represents the local host. Changing this value to a different IP or host name causes the service to fail, since the SNMP service functions only on the local host. The default port number is 1498. You can set a different port for SNMP queries, however, you must ensure that it is not occupied by another SNMP agent.
snmp.usm.user	User name	snmpadmin
snmp.auth.passphrase	Auth password	Sybase4me

3. In the MIB browser, set the **Auth Protocol** to SHA and the **Security Level** to `Auth,NoPriv`.
4. In the object identifier tree of the MIB browser, navigate to `SYBASE-MIB \enterprises\sybase\sup\supObjects\supStatusTable \supStatusEntry` and select **get SNMP variable**.
   Unwired Server status information appears in the data console.

APPENDIX A System Reference

To manage the system effectively, it is crucial to know about Unwired Platform subsystem components and how they fit together. This part outlines many important aspects of the system in quick reference format.

It covers the location of crucial system files and file systems, as well as other reference material that you might need when you are administering the Unwired Platform production environment: for example, logging details, configuration properties, and ports, service names, and processes used by each component.

Installation Directories

Review the Sybase Unwired Platform server component installation directories to ensure a successful installation.

- The following tables show the high-level directories created in a single-node installation (all Unwired Platform server components installed on a single host).
- In a multi-node or cluster installation, some of these directories may not be present on any single host.

By default, Unwired Platform server components are installed in the `C:\Sybase\UnwiredPlatform` directory.

Table 23. Unwired Platform installation subdirectories

Directory	Description
`_jvm`	JVM used by the uninstaller.
`sup`*`XX`*`ebflogs`	Log files created each time `installebf.bat` is run. Appears only in installations upgraded from Unwired Platform 2.0.
`InstallLogs`	Log files created each time the Unwired Platform Runtime installer is used. Use these logs to troubleshoot installer issues.
`JDK`*`x.x.x_x`*	JDK required by Unwired Platform components.
`scc_cert`	Certificate files for Sybase Control Center.

Directory	Description
`Servers`	Unwired Platform server components.
`Servers\Advantage`*xxx*	Device management components for SCC.
`Servers\MessagingServer`	Messaging database server.
`Servers\SQLAnywhere`*xx*	Database server for cache, cluster, and logging databases. Default database file location is the `data\` subdirectory.
`Servers\UnwiredServer`	Unwired Server components.
`Servers\UnwiredServer\doe-c_clu`	Sybase SAP DOE Converter (DOE-C) Command Line Utility components. `CLU.bat` in `bin` directory starts the DOE-C console.
`Servers\UnwiredServer\doecSvlet`	Sybase SAP DOE Converter (DOE-C) runtime components.
`Servers\UnwiredServer\licenses`	SySAM license files. When a license is updated, copy the new files here.
`sup`*XX*`upgrade`	Appears only in installations upgraded from a previous version of Unwired Platform.
`ThirdParty`	License terms of third-party components included in Sybase Unwired Platform.
`Uninstallers`	Unwired Platform Runtime uninstaller.
`Util`	Utilities used by the Unwired Platform Runtime installer.

By default, Sybase Control Center components are installed in the `C:\Sybase\SCC-`*XX* directory.

Note: If you have other Sybase products installed on the same host as Unwired Server, you may have more than one version of Sybase Control Center.

Table 24. Sybase Control Center installation subdirectories

Directory	Description
`auth`	Library files used for related services, such as JAAS.

Directory	Description
`bin`	Scripts to start or stop SCC management framework components.
`common`	Files shared by SCC components.
`conf`	Configuration files, including security providers for administration logins.
`ldap`	LDAP-related files.
`log`	Log files used by SCC and its console plug-ins to capture management framework events only. No Unwired Platform data is captured here, except for administration logins.
`plugins`	Managed resource plug-ins.
`rtlilb`	Runtime library files.
`server`	Class and library files used by the management framework server.
`services`	Class and library files for SCC services.
`shared`	Shared class and library files.
`utility`	Utilities used by SCC.

Port Number Reference

Change Sybase Unwired Platform component port numbers after installation, if necessary.

Proceed with caution when changing port numbers because the change might impact other configuration files that point to that port. You need to be aware of the default Sybase Control Center port numbers so you do not accidentally use these ports when you change Sybase Unwired Platform ports. You can change some Sybase Control Center default ports, but, in some cases, you should not.

Note: To make Unwired Server port number changes, temporarily stop the other service consuming those ports. Use Sybase Control Center to make the changes, then restart Unwired Server.

Port	Description	Default Port	Instructions for Changing
Data tier (CDB) server	Port number for the data tier that manages transactions between the enterprise information system and mobile devices.	5200	Do not change the CDB port.
Management ports	IIOP port number on which the Unwired Server listens for Sybase Control Center administration requests.	2000 2001 for secure management (default)	Default is recommended. No change is required.
Data change notification (DCN)	Port number on which Unwired Server listens for DCN events.	8000 for HTTP 8001 for HTTPS	Configure in Sybase Control Center by expanding the Servers > *<ServerName>* folder and selecting Server Configuration. In the General tab, select the Communication Ports subtab and enter a new DCN port or secure DCN port, as required. See *Security Profiles* in in Sybase Control Center online help.

Port	Description	Default Port	Instructions for Changing
Synchronization	Port numbers on which Unwired Server synchronizes data between the enterprise information system and mobile devices. Messaging port uses a proprietary encryption method, so communication is always encrypted.	2480 for replication 5001 for messaging	Configure in Sybase Control Center by expanding the Servers > *<ServerName>* folder and selecting Server Configuration. In the Replication or Messaging tab, select the Synchronization Listener subtab and enter a new synchronization port, as required. **Note:** If there is a conflict for port 2480 or 2481, Unwired Server will not start, and yo cannot use Sybase Control Center to modify them. To correct the problem, you must temporarily stop the service that uses the conflicting port, then start Unwired Server. For replication payloads, see *Configuring Replication Properties* in Sybase Control Center online help. For messaging payloads, see *Configuring Messaging Properties* in in Sybase Control Center online help.
Advantage Database Server®	Port number for the messaging database.	6262	Sybase recommends that you do not change these ports.

Port	Description	Default Port	Instructions for Changing
Messaging server administration	Port number for the messaging service for Sybase messaging clients.	5100 for administration services	Cannot be changed in Sybase Control Center. Use the `<<UnwiredPlatform_InstallDir>>\Servers\Messaging Server\Bin\AdminWebServicesTool.exe` command line tool to change the messaging service Web service port. This tool has built in online help describing how to use the tool. From the command prompt run: `<UnwiredPlatform_InstallDir>\Servers\Messaging Server\Bin>AdminWebServicesTool.exe set=<port> restart`

Port	Description	Default Port	Instructions for Changing
Sybase Control Center	Additional default port numbers of which to be aware, when modifying port numbers.	9999 for default RMI agent port 2100 for default JMS messaging service port 3638 for default SCC repository database port 8282, 8283 for default Web container ports	• 9999 – default RMI agent port. The port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\RMI\service-config.xml` • 2100 – default JMS messaging service port. The port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\Messaging\service-config.xml` • 3638 – default SCC repository database port. The default port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\SccSADataserver\service-config.xml` • 8282, 8283 – default Web container ports. The default ports are set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\EmbeddedWebContainer\service-config.xml` Before you make any changes to these files, stop Sybase Control Center X.X service. Start the service after you complete the changes. If any of the subsystems fail to start, check the SCC `agent.log` for error messages.

Port	Description	Default Port	Instructions for Changing
Relay server	Port numbers on which Relay Server listens for requests.	80 for the HTTP port 443 for the HTTPS port	Change the value for the cluster in Sybase Control Center for Unwired Platform. You can then generate the file and transfer it to the corresponding Unwired Server host. See *Sybase Control Center > Configure > Configure Unwired Platform > Clusters >Relay Server*.
Unwired Platform reserved	Port numbers reserved for internal use by Unwired Platform components	4343 5500 27000 8002 2638	Do not use these special port numbers for any purpose. These ports differ from Windows reserved ports (1-1023). **Note:** Even if the installer does not detect a conflict at install time, Windows may later use ports in the 1024-64K range for other purposes. Read Microsoft documentation to determine how to reserve Unwired Platform ports. Otherwise, you may experience intermittent problems when starting up platform services due to Windows using these ports for some other purpose at the same time.

Unwired Platform Windows Services

Unwired Platform Windows services run automatically on your host, with many starting up when the host computer is started or when the installation process finishes. Determine what services exist for each runtime component and what dependencies exist among these services.

Depending on the components installed in the cluster you are administering, some of these services may not appear in your list of Windows services; certain data and server components may be installed on different nodes to facilitate redundancy.

Note: Sybase recommends that you only manually start and stop Sybase Unwired Platform services for debugging and troubleshooting purposes.

If you are routinely starting and stopping Unwired Server, you should use Sybase Control Center for that purpose. Sybase Control Center allows you to manage local and remote servers from a single location, and is more efficient than starting and stopping with services or desktop shortcuts.

Component	Service	Description	Dependencies
Unwired Server	Sybase Unwired MessagingDB	The database server that manages the messaging database.	The Sybase Unwired Server service must be started.
	Sybase Unwired CacheDB	The main, or cache, data tier service.	The Sybase Unwired CacheDB service must be started before the Unwired Server service can be started.
	Sybase Unwired SampleDB	The Sybase Unwired Platform sample database.	None.
	Sybase Unwired Server	The Unwired Server service.	Depends on the Sybase Unwired CacheDB service startup and relay server service (if used).
Runtime Databases	Sybase Unwired ClusterDB	The database server that manages the data that supports the operation of the cluster.	This service only appears with the data tier installed on its own server in a cluster; generally it should be started and stopped with the Sybase Unwired CacheDB service. **Note:** A single-node installation runs these databases in the same service (Sybase Unwired CacheDB).
	Sybase Sybase Unwired LogDataDB	The database server that manages logging for Unwired Platform.	This service only appears with the data tier installed on its own server in a cluster; generally it should be started and stopped with the Sybase Unwired CacheDB service.

Component	Service	Description	Dependencies
Sybase Control Center	Sybase Control Center *X.X*	Provides runtime services to manage, monitor, and control distributed Sybase resources. The agent must be running for Sybase Control Center to run.	None.

Processes Reference

Unwired Platform Windows processes vary, depending on your components and license type.

Use this table to determine existing processes for each runtime component.

Component	Service	Processes
Unwired Server	Replication synchronization services (via MobiLink)	Cache database: `dbsrv`*`XX`*`.exe` Synchronization: `mlsrv`*`XX`*`.exe`
	Messaging synchronization services	`AdminWebServices.exe`, `ampservice.exe`, `JMSBridge.exe`, `LBManager.exe`, `OBMO.exe`, `OBServiceManager.exe` Messaging database: `ads.exe`
	Starts MMS service	`mlsrvwrapper.exe`
Relay Server	Relay Server	`rshost.exe`
	RSOE	`rsoe.exe`
Sybase Control Center	Sybase Control Center *X.X*	`sccservice.exe`
	Sybase Control Center repository database	`dbsrv`*`XX`*`.exe`

EIS Data Source Connection Properties Reference

Name and configure connection properties when you create connection pools in Sybase Control Center to enterprise information systems (EIS) .

See also

- *EIS Connection Management Overview* on page 24
- *Viewing and Editing EIS Connection Properties* on page 26

JDBC Properties

Configure Java Database Connectivity (JDBC) connection properties.

This list of properties can be used by all datasource types. Sybase does not document native properties used only by a single driver. However, you can also use native driver properties, naming them using this syntax:

```
<driver_type>:<NativeConnPropName>=<SupportedValue>
```

Note: If Unwired Server is connecting to a database with a JDBC driver, ensure you have copied required JAR files to correct locations. See .

Name	Description	Supported values
After Insert	Changes the value to `into` if a database requires `insert into` rather than the abbreviated `into`.	`into`
Batch Delimiter	Sets a delimiter, for example, a semicolon, that can be used to separate multiple SQL statements within a statement batch.	`<delimiter>`
Blob Updater	Specifies the name of a class that can be used to update database BLOB (long binary) objects when the BLOB size is greater than psMaximumBlobLength.	`<class name>` The class must implement the `com.sybase.djc.sql.BlobUpdater` interface.
Clob Updater	Specifies the name of a class that can be used to update database CLOB (long string) objects when the CLOB size is greater than psMaximumClobLength.	`<class name>` The class must implement the `com.sybase.djc.sql.ClobUpdater` interface.

Name	Description	Supported values
Code Set	Specifies how to represent a repertoire of characters by setting the value of CS_SYB_CHARSET for this datasource. Used when the data in the datasource is localized. If you do not specify the correct code set, characters may be rendered incorrectly.	`[server]` If the value is server, the value of the current application server's defaultCodeSet property is used.
Commit Protocol	Specifies how Unwired Server handles connections for a datasource at commit time, specifically when a single transaction requires data from multiple endpoints. If you use XA, the recovery log is stored in the tx_manager datasource, and its commit protocol must be optimistic. If tx_manager is aliased to another datasource (that is, one that is defined with the aliasFor property), the commit protocol for that datasource must be optimistic. A last-resource optimization ensures full conformance with the XA specification. The commit protocol for all other datasources should be XA_2PC. Alternately, a transaction that accesses multiple datasources for which the commit protocols are optimistic is permitted.	`[optimistic| pessimistic | XA_2PC]` Choose only one of these protocols: • Optimistic – enables connections to be committed without regard for other connections enlisted in the transaction, assuming that the transaction is not marked for rollback and will successfully commit on all resources. Note: if a transaction accesses multiple data sources with commit protocol of "optimistic", atomicity is not guaranteed. • Pessimistic – specifies that you do not expect any multi-resource transactions. An exception will be thrown (and transaction rolled back) if any attempt is made to use more than one "pessimistic" data source in the same transaction. • XA_2PC – specifies use of the XA two phase commit protocol. If you are using two phase commit, then the recovery log is stored in the "tx_manager" data source, and that data source (or the one it is aliased to) must have the commit protocol of "optimistic" or "pessimistic". All other data sources for which atomicity must be ensured should have the "XA_2PC" commit protocol.

Name	Description	Supported values
Datasource Class	Sets the class that implements the JDBC datasource. Use this property (along with the driverClass property) only if you do not have a predefined database-type entry in Unwired Server for the kind of SQL database you are connecting to. For example, you must use this property for MySQL database connections. You can implement a datasource class to work with a distributed transaction environment. Because Unwired Server supports distributed transactions, some datasources may require that a datasource class be implemented for Unwired Server to interact with it. For two-phase transactions, use the xaDataSourceClass connection property instead.	`<com.`*`mydata-source`*`.jdbc.Driver>`
Database Command Echo	Echoes a database command to both the console window and the server log file. Use this property to immediately see and record the status or outcome of database commands. When you enable this property, Unwired Server echoes every SQL query to `ml.log`, which may help you debug your application.	`[true|false]` Set a value of 1 to echo the database commands like `databaseStartCommand`, and `databaseStopCommand`. Otherwise, do not set this property, or use a value of 0 to disable the echo.

Name	Description	Supported values
Database Create Command	Specifies the operating system command used to create the database for this datasource. If this command is defined and the file referenced by ${databaseFile} does not exist, the command is run to create the database when an application component attempts to obtain the first connection from the connection pool for this datasource.	`<command>` Example: `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32\dbinit -q ${databaseFile}`
Database File	Indicates the database file to load when connecting to a datasource. Use this property when the path to the database file differs from the one normally used by the database server. If the database you want to connect to is already running, use the databaseName connection parameter.	`<string>` Supply a complete path and file name. The database file you specify must be on the same host as the server.

Name	Description	Supported values
Database Name	Identifies a loaded database with which to establish a connection, when connecting to a datasource. Set a database name, so you can refer to the database by name in other property definitions for a datasource. If the database to connect to is not already running, use the databaseFile connection parameter so the database can be started. **Note:** For Unwired Server, you typically do not need to use this property. Usually, when you start a database on a server, the database is assigned a name. The mechanism by which this occurs varies. An administrator can use the DBN option to set a unique name, or the server may use the base of the file name with the extension and path removed.	`[DBN|default]` If you set this property to default, the name is obtained from the DBN option set by the database administrator. If no value is used, the database name is inherited from the database type.
Database Start Command	Specifies the operating system command used to start the database for this datasource. If this command is defined and the database is not running, the command is run to start the database when the datasource is activated.	`<command>` Example: `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32\dbsrv`*`XX`*`.exe`

Database Stop Command	Specifies the operating system command used to stop the database for this datasource. If this property is defined and the database is running, this command executes during shutdown.	`<command>`
		
		For a SQL Anywhere® database, where the user name and password are the defaults (dba and sql), enter:
		
		`<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32\dbsrv`*`XX`*`.exe`

Name	Description	Supported values
Database Type	Specifies the database type.	`<database type>`
Database URL	Sets the JDBC URL for connecting to the database if the datasource requires an Internet connection. Typically, the server attempts to construct the database URL from the various connection properties you specify (for example, portNumber, databaseName). However, because some drivers require a special or unique URL syntax, this property allows you to override the server defaults and instead provide explicit values for this URL.	`<JDBCurl>` The database URL is JDBC driver vendor-specific. For details, refer to the driver vendor's JDBC documentation.
Driver Class	Sets the name of the class that implements the JDBC driver. Use this property (along with the dataSourceClass property) only if you do not have a predefined database-type entry in Unwired Server for the kind of SQL database you are connecting to. For example, MySQL database connections require you to use this connection property. To create a connection to a database system, you must use the compatible JDBC driver classes. Sybase does not provide these classes; you must obtain them from the database manufacturer.	*`<Class.forName("foo.bar.Driver")>`* Replace <Class.forName("foo.bar.Driver")> with the name of your driver.
Driver Debug	Enables debugging for the driver.	`[true|false]` Set to true to enable debugging, or false to disable.
Driver Debug Settings	Configures debug settings for the driver debugger.	`[default|<setting>]` The default is STATIC:ALL.

Name	Description	Supported values
Initial Pool Size	Sets the initial number of connections in the pool for a datasource. In general, holding a connection causes a less dramatic performance impact than creating a new connection. Keep your pool size large enough for the number of concurrent requests you have; ideally, your connection pool size should ensure that you never run out of available connections. The initialPoolSize value is applied to the next time you start Unwired Server.	`<int>` Replace <int> with an integer to preallocate and open the specified number of connections at start-up. The default is 0. Sybase suggests that you start with 0, and create additional connections as necessary. The value you choose allows you to create additional connections before client synchronization requires the server to create them.
Is Download Zipped	Specifies whether the driver file downloaded from jdbcDriverDownloadURL is in `.ZIP` format. This property is ignored if the value of jdbcDriverDownloadURL connection is an empty string.	`[True|False]` The default is false. The file is copied, but not zipped to `<UnwiredPlatform-install>\lib\jdbc`. Set isDownloadZipped to true to save the file to `<UnwiredPlatform-install>\lib\jdbc` and unzip the archived copy.
JDBC Driver Download URL	Specifies the URL from which you can download a database driver. Use this property with isDownloadZipped to put the driver in an archive file before the download starts.	`<URL>` Replace <URL> with the URL from which the driver can be downloaded.

Name	Description	Supported values
Language	For those interfaces that support localization, this property specifies the language to use when connecting to your target database. When you specify a value for this property, Unwired Server: • Allocates a CS_LOCALE structure for this connection • Sets the CS_SYB_LANG value to the language you specify • Sets the Microsoft SQL Server CS_LOC_PROP connection property with the new locale information Unwired Server can access Unicode data in an Adaptive Server® 12.5 or later, or in Unicode columns in Adaptive Server 12.5 or later. Unwired Server automatically converts between double-byte character set (DBCS) data and Unicode, provided that the Language and CodeSet parameters are set with DBCS values.	`<language>` Replace <language> with the language being used.
Max Idle Time	Specifies the number of seconds an idle connection remains in the pool before it is dropped.	`<int>` If the value is 0, idle connections remain in the pool until the server shuts down. The default is 60.

Name	Description	Supported values
Max Pool Size	Sets the maximum number of connections allocated to the pool for this datasource. Increase the maxPoolSize property value when you have a large user base. To determine whether a value is high enough, look for ResourceMonitorTimeoutException exceptions in `<hostname>-server.log`. Continue increasing the value, until this exception no longer occurs. To further reduce the likelihood of deadlocks, configure a higher value for maxWaitTime. To control the range of the pool size, use this property with minPoolSize.	`<int>` A value of 0 sets no limit to the maximum connection pool size. The default is 10.
Max Wait Time	Sets the maximum number of seconds to wait for a connection before the request is cancelled.	`<int>` The default is 60.
Max Statements	Specifies the maximum number of JDBC prepared statements that can be cached for each connection by the JDBC driver. The value of this property is specific to each JDBC driver.	`<int>` A value of 0 (default) sets no limit to the maximum statements.
Min Pool Size	Sets the minimum number of connections allocated to the pool for this datasource.	`<int>` A value of 0 (default) sets no limit to the minimum connection pool size.

Name	Description	Supported values
Network Protocol	Sets the protocol used for network communication with the datasource. Use this property (along with the driverClass, and dataSourceClass properties) only if you do not have a predefined database-type entry in Unwired Server for the kind of SQL database you are connecting to. For example, you may be required to use this property for MySQL database connections.	The network protocol is JDBC driver vendor-specific. There are no predefined values. See the driver vendor's JDBC documentation.
Password	Specifies the password for connecting to the database.	`[default|<password>]`
Ping and Set Session Auth	Runs the ping and session-authorization commands in a single command batch; may improve performance. You can only enable the Ping and Set Session Auth property if you have enabled the Set Session Auth property so database work runs under the effective user ID of the client.	`[True|False]` Set to true to enable, or false to disable.
Ping Connections	Pings connections before attempting to reuse them from the connection pool.	`[True|False]` Set to true to enable ping connections, or false to disable.
Ping SQL	Specify the SQL statement to use when testing the database connection with ping.	`[default|<statement>]` Replace <statement> with the SQL statement identifier. The default is "select 1".
Port Number	Sets the server port number where the database server listens for connection requests.	`[default|<port>]` Replace <port> with the TCP/IP port number to use (that is, 1 – 65535). If you set the value as default, the default protocol of the datasource is used.

Name	Description	Supported values
PS Maximum Blob Length	Indicates the maximum number of bytes allowed when updating a BLOB datatype using PreparedStatement.setBytes.	`[default|<int>]` Replace <int> with the number of bytes allowed during an update. The default is 16384.
PS Maximum Clob Length	Indicates the maximum number of characters allowed when updating a CLOB datatype using PreparedStatement.setString.	`[default|<int>]` Replace <int> with the number of bytes allowed during an update. The default is 16384.
Role Name	Sets the database role that the user must have to log in to the database.	`[default|<name>]` If you set this value to default, the default database role name of the datasource is used.
Server Name	Defines the host where the database server is running.	`<name>` Replace <name> with an appropriate name for the server.
Service Name	Defines the service name for the datasource. For SQL Anywhere servers, use this property to specify the database you are attaching to.	`<name>` Replace <name> with an appropriate name for the service.
Set Session Auth	Establishes an effective database identity that matches the current mobile application user. If you use this property, you must also use setSessionAuthSystemID to set the session ID. Alternately you can pingAndSetSessionAuth if you are using this property with pingConnection. The pingAndSetSessionAuth property runs the ping and session-authorization commands in a single command batch, which may improve performance.	`[true|false]` Choose a value of 1 to use an ANSI SQL set session authorization command at the start of each database transaction. Set to 0 to use session-based authorizations.

Name	Description	Supported values
Set Session Auth System ID	If Set Session Authorization is enabled, specifies the database identity to use when the application server accesses the database from a transaction that runs with "system" identity.	`<database identity>` Replace <database identity> with the database identifier.
Start Wait	Sets the wait time (in seconds) before a connection problem is reported. If the start command completes successfully within this time period, no exceptions are reported in the server log. startWait time is used only with the databaseStartCommand property.	`<int>` Replace <int> with the number of seconds Unwired Server waits before reporting an error.
Truncate Nanoseconds	Sets a divisor/multiplier that is used to round the nanoseconds value in a java.sql.Timestamp to a granularity that the DBMS supports.	`[default|<int>]` The default is 10 000 000.
Use Quoted Identifiers	Specifies whether or not SQL identifiers are quoted.	`[True|False]` Set to true to enable use of quoted identifiers, or false to disable.
User	Identifies the user who is connecting to the database.	`[default|<user name>]` Replace <user name> with the database user name.
XA Datasource Class	Specifies the class name or library name used to support two-phase commit transactions, and the name of the XA resource library.	`<class name>` Replace <class name> with the class or library name. • SQL Anywhere database: `com.sybase.jdbc3.jdbc.SybXADataSource` • Oracle database: `oracle.jdbc.xa.client.OracleXADataSource`

See also

- *EIS Connection Management Overview* on page 24
- *Changing the Cache Database Server Pool Size* on page 22

SAP Java Connector Properties

Configure SAP Java Connector (JCo) connection properties.

For a comprehensive list of SAP JCo properties you can use to create an instance of a client connection to a remote SAP system, see *http://help.sap.com/javadocs/NW04/current/jc/com/sap/mw/jco/JCO.html#createClient(java.util.Properties)*.

Table 25. General connection parameters

Name	Description	Supported values
Client Number	Specifies the SAP client.	Three-digit client number; preserve leading zeros if they appear in the number
Logon User	Specifies the login user ID.	User name for logging in to the SAP system If using X.509 certificate authentication, remove the JCo properties `jco.client.passwd` and `jco.client.user` defined for the SAP connection profile in Sybase Control Center (SCC).
Password	Specifies the login password.	Password for logging in to the SAP system
Language	Specifies a login language.	ISO two-character language code (for example, EN, DE, FR), or SAP-specific single-character language code. As a result, only the first two characters are ever used, even if a longer string is entered. The default is EN.
System Number	Indicates the SAP system number.	SAP system number
Host Name	Identifies the SAP application server.	Host name of a specific SAP application server
Message Server	Identifies the SAP message server.	Host name of the message server

Name	Description	Supported values
Gateway Host	Identifies the SAP gateway host.	Host name of the SAP gateway Example: `GWHOST=hs0311`
Gateway Service	Identifies the SAP gateway service.	Service name of the SAP gateway Example: `GWSERV=sapgw53`
R/3 Name	Specifies R/3 name.	Name of the SAP system
Server Group	Identifies the group of SAP application servers.	Group name of the application servers
External Server Program	Identifies the program ID of the external server program.	Path and name of the external RFC server program, or program ID of a registered RFC server program Example: `TPNAME=/sap/srfcserv`
External Server Program Host	Identifies the host of the external server program. This information determines whether the RFC client connects to an RFC server started by the SAP gateway or to an already registered RFC server. **Note:** If the gateway host and external server program host are different, make sure that the SAP gateway has access to start the server program through a remote shell.	Host name of the external RFC server program Example: `TPHOST=hs0311`
Remote Host Type	Identifies the type of remote host.	2: R/2 3: R/3 E: external
RFC Trace	Specifies whether or not to enable RFC trace.	0: disable 1: enable

Name	Description	Supported values
Initial Codepage	Identifies the initial code page in SAP notation. A code page is used whenever character data is processed on the application server, appears on the front end, or is rendered by a printer.	Four-digit SAP code page number
Enable ABAP Debugging	Enables or disables ABAP debugging. If enabled, the connection is opened in debug mode and the invoked function module can be stepped through in the debugger. For debugging, an SAP graphical user interface (SAPGUI) must be installed on the same machine the client program is running on. This can be either a normal Windows SAPGUI or a Java GUI on Linux/UNIX systems.	0: no debugging 1: attach a visible SAPGUI and break at the first ABAP statement of the invoked function module
Remote GUI	Specifies whether a remote SAP graphical user interface (SAPGUI) should be attached to the connection. Some older BAPIs need an SAPGUI because they try to send screen output to the client while executing.	0: no SAPGUI 1: attach an "invisible" SAPGUI, which receives and ignores the screen output 2: attach a visible SAPGUI For values other than 0 a SAPGUI needs to be installed on the machine, where the client program is running. This can be either a Windows SAPGUI or a Java GUI on Linux/Unix systems.
Get SSO Ticket	Generates an SSO2 ticket for the user after login to allow single sign-on. If RfcOpenConnection() succeeds, you can retrieve the ticket with RfcGetPartnerSSOTicket() and use it for additional logins to systems supporting the same user base.	0: do not generate SSO2 ticket 1: generate SSO2 ticket

Name	Description	Supported values
Use Cookie Version 2	Indicates whether or not to use the specified SAP Cookie Version 2 (SSO2) as the login ticket instead of user ID and password.	User: $MYSAPSSO2$ Password: Base64-encoded ticket Login with single sign-on is based on secure network connection (SNC) encryption and can only be used in combination with an SNC.
Use X509	Indicates whether or not to use the specified X509 certificate as the login certificate instead of user ID and password.	User: $X509CERT$ Password: Base64-encoded ticket Login with X509 is based on secure network connection (SNC) encryption and can only be used in combination with an SNC.
Logon Check	Enables or disables login check at open time.	0: disable 1: enable If you set this to 0, RfcOpenConnection() opens a network connection, but does not perform the login procedure. Therefore, no user session is created inside the back-end system. This parameter is intended only for executing the function module RFC_PING.
Additional GUI Data	Provides additional data for graphical user interface (GUI) to specify the SAProuter connection data for the SAPGUI when it is used with RFC.	/H/ *router string* : the entire router string for the SAPGUI /P/ *password* : specify this value if the password for the SAPGUI connection is not the same as the password for the RFC connection.
GUI Redirect Host	Identifies which host to redirect the remote graphical user interface to.	Host name
GUI Redirect Service	Identifies which service to redirect the remote graphical user interface to.	Name of the service
Remote GUI Start Program	Indicates the program ID of the server that starts the remote graphical user interface.	Program ID of the server

Name	Description	Supported values
SNC Mode	Enables or disables secure network connection mode.	0: off 1: on
SNC Partner	Identifies the secure network connection partner.	Secure network connection name of the application server (for example, p:CN=R3, O=XYZ-INC, C=EN)
SNC Level	Specifies the secure network connection security level.	1: digital signature 2: digital signature and encryption 3: digital signature, encryption, and user authentication 8: default value defined by backend system 9: maximum value that the current security product supports
SNC Name	Indicates the secure network connection name. This property overrides the default secure network connection partner.	Token or identifier representing the external RFC program
SNC Service Lib Path	Identifies the path to the SAP crytpographic library that provides secure network connection service.	Full path and name of third-party security library. You must download and install the library from the SAP Service Marketplace.
R/2 Destination	Identifies a configured R/2 system defined in the sideinfo configuration.	
Logon ID	Defines the string for SAPLOGON on 32-bit Windows.	String key to read parameters from the saplogon.ini file created by the SAPLogon GUI program on Windows
External Authentication Data	Provides data for external authentication (PAS). This is an old login mechanism similar to SSO; Sybase recommends that you do not use this approach.	
External Authentication	Specifies type of external authentication (PAS). See External Authentication Data property.	

See also

- *EIS Connection Management Overview* on page 24

SAP DOE-C Properties

Configure SAP Data Orchestration Engine Connector (DOE-C) properties. This type of connection is available in the list of connection templates only when you deploy a DOE-C package. No template exists for these types of connections.

Note: If you change the username or password property of a DOE-C connection, you must reopen the same dialog and click `Test Connection` after saving. Otherwise the error state of this DOE-C package is not set properly, and an error message is displayed. This will not work if you click `Test Connection` before saving the properties.

Name	Description	Supported values
Username	Specifies the SAP user account ID. The SAP user account is used during interaction between the connected SAP system and client for certain administrative activities, such as sending acknowledgment messages during day-to-day operations or "unsubscribe" messages if a subscription for this connection is removed. This account is not used for messages containing business data; those types of messages are always sent within the context of a session authenticated with credentials provided by the mobile client. The technical user name and password or certificateAlias must be set to perform actions on subscriptions. The certificateAlias is mutually exclusive with and overrides the technical user name and password fields if set. The technical user name and password fields can be empty, but only if `certificateAlias` is set.	Valid SAP login name for the DOE host system.
Password	Specifies the password for the SAP user account.	Valid password.

Name	Description	Supported values
DOE SOAP Timeout	Specifies a timeout window during which unresponsive DOE requests are aborted.	Positive value (in seconds). The default is 420 (7 minutes).
DOE Extract Window	Specifies the number of messages allowed in the DOE extract window.	Positive value (in messages). The default is 50. When the number of messages in the DOE extract window reaches 50% of this value, DOE-C sends a `StatusReqFromClient` message, to advise the SAP DOE system of the client's messaging status and acknowledge the server's state. The default value is 50.
Packet Drop Size	Specifies the size, in bytes, of the largest JavaScript Object Notation (JSON) message that the DOE connector processes on behalf of a JSON client. The packet drop threshold size should be carefully chosen, so that it is larger than the largest message sent from the DOE to the client, but smaller than the maximum message size which may be processed by the client.	Positive value (in bytes). The default is 1048576 bytes (1MB). Do not set lower than 4096 bytes; there is no maximum limitation.
Service Address	Specifies the DOE URL.	Valid DOE URL. If you are using DOE-C with SSO: • Modify the port from the standard `http://host:8000` to `https://host:8001/`. • Add the certificate being used as the technical user and DOE-C endpoint security profile certificate to the SAP DOE system's SSL Server certificate list by using the STRUST transaction. See your SAP documentation for details.

Name	Description	Supported values
Listener URL	Specifies the DOE-C server listener URL.	Valid DOE-C listener URL, for example `http://<sup_host-name>:8000/doe/publish`.
SAP Technical User Certificate Alias	Sets the alias for the Unwired Platform keystore entry that contains the X.509 certificate for Unwired Server's SSL peer identity. If you do not set a value, mutual authentication for SSL is not used when connecting to the Web service. If you are using DOE-C with SSO use the "SAP Technical User Certificate Alias" only for configurations which require the technical user to identify itself using an X.509 certificate; it specifies the Certificate Alias to be used as the technical user. This overrides the "Username" and "Password" settings normally used.	Valid certificate alias.
Login Required	Indicates whether authentication credentials are required to login. The default value is true. For upgraded packages, "login-required=false" gets converted to "login-required=true" and a No-Auth security configuration "DOECNoAuth" is assigned to the upgraded package.	A read-only property with a value of true.

See also

- *EIS Connection Management Overview* on page 24

Proxy Properties

(Applies only to Online Data Proxy) Proxy properties identify the application endpoint and the pool size.

Name	Description	Supported values
User	Not currently used	

Name	Description	Supported values
Certificate Alias	Not currently used	
Address	Corresponds to the Application end-point provided at the time of registering an application.	Must be a valid application end-point.
Pool Size	Determines the maximum number of connections allocated to the pool for this datasource.	The default value set for the pool size is 25.
Password	Not currently used	

Note: When the application end-point for a registered application is modified under the **Applications** node, you have to manually update the **Address** in the proxy properties of the connection pool.

See also

- *EIS Connection Management Overview* on page 24

Web Services Properties

Configure connection properties for the Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) architectures.

Name	Description	Supported Values
Password	Specifies the password for HTTP basic authentication, if applicable.	Password
Address	Specifies a different URL than the port address indicated in the WSDL document at design time.	HTTP URL address of the Web service
User	Specifies the user name for HTTP basic authentication, if applicable.	User name
Certificate Alias	Sets the alias for the Unwired Platform keystore entry that contains the X.509 certificate for Unwired Server's SSL peer identity. If you do not set a value, mutual authentication for SSL is not used when connecting to the Web service.	Use the alias of a certificate stored in the Unwired Server certificate keystore.

See also

- *EIS Connection Management Overview* on page 24

Command Line Utilities

Invoke command line utilities directly from the operating system.

Unwired Platform includes a set of command line utilities for carrying out routine administrative tasks, such as deploying a package or maintaining a cache database (CDB). You can also include these commands in batch files for repeated use.

To launch a utility:

- Double-click its icon, if it has one.
- If it does not have an icon in the program group, enter the utility program command at the Windows command prompt.

Relay Server Utilities

Use relay server utilities to administer the relay server: rshost and regRelayServer.

Launch these utilities from the command line; they are not available from any other administration tool.

Relay Server Host (rshost) Utility

State Manager (rshost) maintains state information across Relay Server client requests and RSOE sessions, and manages the Relay Server log file. It also provides a command line utility on the Relay Server host, to update the Relay Server configuration and archive the Relay Server log.

Syntax

Variable declaration:

```
rshost [option]+
```

Parameters

- <***option***> **–**

Option	Description
`-f <filename>`	Relay Server configuration file
`-o <filename>`	Relay Server log file
`-oq`	Prevents popup window on startup error
`-q`	Run in minimized window

Option	Description
`-qc`	Close window on completion
`-u`	Update Relay Server configuration
`-ua`	Archive Relay Server log file to `<yymmdd><nn>.log` and truncate

Usage

For more information, see "Relay Server State Manager command line syntax" in *SQL Anywhere documentation*.

Register Relay Server (regRelayServer) Utility

Use `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin\regRelayServer.bat` to perform multiple relay server and RSOE administrative actions that can also be performed in Sybase Control Center.

Syntax

```
regRelayServer [config | export | exportRSconfig | migrate|
quickconfig | remove | start | stop]
```

Parameters

- **config –** Configure relay server or an RSOE for an Unwired Server node with a named XML file as input. Supported options include:
  - **-f <*filename*> –** The input file and path.
- **export –** Export the existing RSOE configuration into a file. Supported options include:
  - **-o <*filename*> –** The output file and path.
- **exportRSconfig –** Export the actual relay server configuration properties file, which could be used by the Unwired Platform administrator to configure Unwired Server farms, server nodes, for remaining relay servers that require configuration or updates. Supported options include:
  - **-o <*filename*> –** The output file and path.
  - **-h <*name*> –** (Required) The name of the host.
  - **-p <*port*> –** (Required) The port number used by the relay server.
- **migrate –** Migrate the configuration values defined in the 1.5.5 version of `<UnwiredPlatfrom_InstallDir>\Servers\UnwiredServer\config\relayserver.properties` file and port them to the 1.6 `rsconfig.xml` file. Be aware that:

  - The `relayserver.properties` file must exist and must use supported values for the relay server and RSOE. Ensure that no changes to the file occur while the file is being processed by this command.
  - This command is only valid until `relayserver.properties` has been migrated for a specific relay server host.
  - Once you migrate `relayserver.properties`, do not update it.
- **quickconfig –** Rapidly create a relay server configuration, collecting only connection property values and use system-generate defaults for the remaining properties required. RSOEs deployed with this method require manual property edits before they can be started. Export the configuration and update the properties in the file. You can then re-apply the configuration with the configure command.
- **remove –** Deletes one or all RSOE process for the Unwired Server node. However, the configuration file is not deleted, because other RSOEs on the server node may require it. Supported options include:

  - **-f <*name*> –** The relay server farm name.
  - **-h <*name*> –** The name of the relay server host.
  - **-p <*port*> –** (Required) The port number used by the relay server.

  If you do not specify -h, -p, and -f options, all RSOEs for the Unwired Server node are removed. Otherwise, only those identified are removed.
- **start –** Start one or all RSOE processes for the Unwired Server node. Supported options include:

  - **-f <*name*> –** The relay server farm name.
  - **-h <*name*> –** The name of the relay server host.
  - **-p <*port*> –** (Required) The port number used by the relay server.

  If you do not specify -h, -p, and -f options, all RSOEs for the Unwired Server node start. Otherwise, only those identified are started.
- **stop –** Stop one or all RSOE processes for the Unwired Server node. Supported options include:

  - **-f <*name*> –** The relay server farm name.
  - **-h <*name*> –** The name of the Unwired Server host.
  - **-p <*port*> –** (Required) The port number used by the Unwired Server.

  If you do not specify -h, -p, and -f options, all RSOEs for the Unwired Server node stop. Otherwise, only those identified are stopped.

Examples

- **Export properties then configure a new relay server –** In this example, an administrator uses this utility multiple times but on different host computers:

1. The administrator exports the file for a relay server deployed in a test environment. This configuration is stable and the administrator now wishes to propagate these properties to all new relay servers in a production environment. The administrator runs this command with the on the test host computer:
   ```
   regrelayserver.bat exportRSconfig -o <path>\rsconfig.xml
   ```
2. The administrator transfers the file to the new host. The command used to apply the file on the new host is:
   ```
   regrelayserver.bat config -f <path>\rsconfig.xml
   ```

- **Registering a new RSOE and configuring it with rsoe.config.template.xml –** In this example an administrator has copied and modified `<UnwiredPlatfrom_InstallDir>\Servers\UnwiredServer\config\rsoeConfig.template.xml.`. Because the XML elements do not include an `ID` attribute value, this utility will treat the RSOE as newly deployed, and register it in the cluster database in addition to configuring it. The administrator uses this command:
  ```
  regrelayserver.bat config -f <path>\rsoe.config.template.xml
  ```
- **Start all RSOEs on the Unwired Server node –** Once RSOEs are configured, rather than restarting the host computer, the administrator starts all newly configured RSOEs on the current machine with this command:
  ```
  regrelayserver.bat start
  ```
- **Stop only RSOEs of a named relay server farm –** An administrator needs to make a change to one of the RSOEs recently started, and uses this command:
  ```
  regrelayserver.bat stop -f myhost.MBS.farm -h myUServerhost -p
  5001
  ```

Usage

Ensure the Unwired Server node cluster database is running before you use this utility.

RSOE Service (rsoeservice) Utility

Installs, removes, starts, and stops the relay server outbound enabler (RSOE) as a Windows service from the command line. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin`.

Use this utility to manage RSOE services. To apply the changes in `relayserver.properties`, use **regRelayServer.bat**.

Syntax

```
rsoeservice [install auto | install manual | remove | start | stop]
```

Parameters

- **install auto –** installs RSOE as a windows service in auto-start mode.
- **install manual –** installs RSOE as a windows service in manual start mode.

- **remove –** uninstalls the RSOE service.
- **start –** starts the RSOE service.
- **stop –** stops the RSOE service.

Examples

- **Basic Example –** This command installs RSOE as an auto-start Windows service:

```
rsoeservice install auto
```

Certificate and Key Management Utilities

Use the certificate management utilities to encrypt Unwired Server ports.

Launch these utilities from the command line; the certificate and key management utilities are not available from any other administration tool.

Use **createcert** and **createkey** for MobiLink and Ultralite server/client purposes (specific for replication payload protocol packages). For all other purposes, use **keytool** or the PKI system deployed to your environment.

Certificate Creation (createcert) Utility

Generates X.509 certificates or signs pregenerated certificate requests. This utility is located in <*UnwiredPlatform_InstallDir*>\Servers\SQLAnywhere*XX*\BIN*XX*.

You may choose to purchase certificates from a third party. These certificate authorities (CAs) provide their own tools for creating certificates. You can use **createcert** to create certificates for development and testing; you can also use it for production certificates.

Syntax

```
createcert [options]
```

Parameters

- **[*options*] –** these options are available through the **createcert** utility:

Option	Description
`-r`	Creates a PKCS #10 certificate request. **createcert** does not prompt for a signer or any other information used to sign a certificate.
`-s <filename>`	Signs the PKCS #10 certificate request that is in the specified file. The request can be DER or PEM encoded. **createcert** does not prompt for key generation or subject information.

Note: To create a signed certificate, use **createcert** without options. To break the process into two separate steps, for example so one person creates a request and another person

signs it, the first person can run **createcert** with `-r` to create a request and the second person can sign the request by running **createcert** with `-s`.

When you run **createcert**, you are asked for all or some of this information, depending on whether you specified `-r`, `-s`, or neither.

- `Choose encryption type` – select RSA.
- `Enter RSA key length (512-16384)` – this prompt appears only if you chose RSA encryption. Specify a length between 512 bits and 16384 bits.
- Subject information – enter this information, which identifies the entity:
  - `Country Code`
  - `State/Province`
  - `Locality`
  - `Organization`
  - `Organizational Unit`
  - `Common Name`
- (Optional) `Enter file path of signer's certificate` – supply a location and file name for the signer's certificate. If you supply this information, the generated certificate is a signed certificate. If you do not supply this information, the generated certificate is a self-signed root certificate.
- `Enter file path of signer's private key` – supply a location and file name to save the private key associated with the certificate request. This prompt appears only if you supplied a file in the previous prompt.
- `Enter password for signer's private key` – supply the password that was used to encrypt the signer's private key. Supply this password only if the private key was encrypted.
- (Optional) `Serial number` – supply a serial number. The serial number must be a hexadecimal string of 40 digits or less. This number must be unique among all certificates signed by the current signer. If you do not supply a serial number, **createcert** generates a GUID as the serial number.
- `Certificate will be valid for how many years (1-100)` – specify the number of years for which the certificate is valid. After this period, the certificate expires, along with all certificates it signs.
- `Certificate Authority (y)es or (n)o` – indicate whether this certificate can be used to sign other certificates. The default value is no.
- `Key usage` – supply a comma-separated list of numbers that indicate the ways in which the certificate's private key can be used. The default, which depends on whether the certificate is a certificate authority, should be acceptable for most situations.
- `File path to save request` – this prompt appears only if you specify the -r option. Supply a location and file name for the PCKS #10 certificate request. Supply a location and file name in which to save the certificate. The certificate is not saved unless you specify a location and file name.

- `Enter file path to save private key` – supply a location and file name in which to save the private key. Enter a password to protect private key. Optionally, supply a password with which to encrypt the private key. If you do not supply a password, the private key is not encrypted. This prompt appears only if you supplied a file in the previous prompt.
- `Enter file path to save identity` – supply a location and file name in which to save the identity. The identity file is a concatenation of the certificate, signer, and private key. This is the file that you supply to the server at start-up. If the private key was not saved, **createcert** prompts for a password to save the private key. Otherwise, it uses the password provided earlier. The identity is not saved unless you provide a file name. If you do not save the identity file, you can manually concatenate the certificate, signer, and private key files into an identity file.

Examples

- **Example 1: –** creates a self-signed certificate. No file name is provided for the signer's certificate, which makes it a self-signed root certificate.

```
<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers
\SQLAnywhereXX\BINXX>createcert
SQL Anywhere X.509 Certificate Generator Version xx.xx.xx.xx
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
State/Province: CA
Locality: Dublin
Organization: MyCompanyCA
Organizational Unit: PTO
Common Name: MyCompanyCA
Enter file path of signer's certificate:
Certificate will be a self-signed root
Serial number [generate GUID]:<enter>
Generated serial number: 3f52ee68c8604e48b8359e0c0128da5a
Certificate valid for how many years (1-100): 10
Certificate Authority (Y/N) [N]: Y
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [6,7]: <enter>
Enter file path to save certificate: rsa_root.crt
Enter file path to save private key: rsa_key.key
Enter password to protect private key: <MyPwd>
Enter file path to save identity: id.pem
```

- **Example 2: Generating an enterprise root certificate –** to generate an enterprise root certificate (a certificate that signs other certificates), create a self-signed root certificate

with a CA. The procedure is similar to Example 1. However, the response to the CA prompt should be `yes` and choice for roles should be option `6,7` (the default).

```
Certificate Authority (Y/N) [N]: y
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [6,7]: 6,7
```

Key Creation (createkey) Utility

Creates an RSA key pairs for use with Unwired Server end-to-end encryption. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhereXX\BINXX`.

Syntax

```
createkey
```

When you run **createkey**, you are prompted for this information:

- `Choose encryption type` – choose RSA.
- `Enter RSA key length (512-16384)` – this prompt appears only if you chose RSA encryption. Choose a length between 512 bits and 16384 bits.
- `Enter file path to save public key` – specify a file name and location for the generated PEM-encoded public key. This file is specified on the MobiLink client by the e2ee_public_key protocol option.
- `Enter file path to save private key` – specify a file name and location for the generated PEM-encoded private key. This file is specified on the MobiLink server via the e2ee_private_key protocol option.
- `Enter password to protect private key` – optionally, supply a password with which to encrypt the private key. The private key is not encrypted if you do not supply a password. This password is specified on the MobiLink server via the e2ee_private_key_password protocol option.

Examples

- **Example –** creates an RSA key pair:

```
>createkey
SQL Anywhere Key Pair Generator Version 11.0.0.2376
Enter RSA key length (512-16384): 2048
Generating key pair...
Enter file path to save public key: rsa_key_public.key
```

```
Enter file path to save private key: rsa_key_private.key
Enter password to protect private key: pwd
```

Key Tool (keytool) Utility

keytool is a JDK utility used to manage a keystore (database) of private keys and associated X.509 certificates, as well as certificates from trusted entities. **keytool** is in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\SQLAnywhere11\Sun\JRE160_x86\bin`.

keytool enables users to create and manage their own public/private key pairs and associated certificates for use in self-authentication or data integrity and authentication services, using digital signatures. It also allows users to cache the public keys (in the form of certificates) of their communicating peers.

Syntax

```
keytool -list | -printcert | -import | -export| -delete | -selfcert |
-certreq | -genkey [options]
```

Parameters

- **-list –** displays the contents of a keystore or keystore entry.
- **-printcert –** displays the contents of a certificate stored in a file. Check this information before importing a certificate as a trusted certificate. Make sure certificate prints as expected.
- **-import –** imports:
  - a certificate or certificate chain to the list of trusted certificates, or,
  - a certificate reply received from a certificate authority (CA) as the result of submitting a certificate signing request (CSR).

  The value of the `-alias` option indicates the type of import you are performing. If the alias exists in the database, then it is assumed you want to import a certificate reply. **keytool** checks whether the public key in the certificate reply matches the public key stored with the alias, and exits if they do not match. If the alias identifies the other type of keystore entry, the certificate is not imported. If the alias does not exist, it is created, and associated with the imported certificate.
- **-export –** exports a certificate to a file.
- **-delete –** deletes a certificate from the list of trusted certificates.
- **-selfcert –** generates a self-signed certificate. The generated certificate is stored as a single-element certificate chain in the keystore entry identified by the specified alias, where it replaces the existing certificate chain.
- **-certreq –** generates a certificate signing request (CSR), using the PKCS #10 format. A CSR is intended to be sent to a CA, which authenticates the certificate requestor and returns a certificate or certificate chain, used to replace the existing certificate chain in the keystore.

- **-genkey –** generates a key pair (a public key and associated private key). Wraps the public key into an X.509 v1 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are stored in a new keystore entry identified by `<alias>`.

-genkey Option	Description
`-keystore <keyStoreLocation>`	Name and location of the persistent keystore file for the keystore managed by **keytool**. If you specify, in the `-keystore` option, a keystore that does not exist, a keystore is created. If you do not specify a `-keystore` option, the default keystore is a file named `.keystore` in your home directory. If that file does not exist, it is created.
`-storepass <password>`	The password protecting keystore integrity. The password must be at least 6 characters long and provided to all commands that access the keystore contents. For such commands, if a `-storepass` option is not provided at the command line, the user is prompted for it.
`-file <certificateFile>`	The certificate file location.
`-noprompt`	During import, removes interaction with the user.
`-trustcacerts`	When importing a certificate reply, the certificate reply is validated using trusted certificates from the keystore, and using the certificates configured in the `cacerts` keystore file. This file resides in the JDK security properties directory, `java.home\lib\security`, where java.home is the runtime environment's directory. The `cacerts` file represents a system-wide keystore with CA certificates. System administrators can configure and manage that file using **keytool**, specifying "jks" as the keystore type.
`-alias <alias>`	The logical name for the certificate you are using.
`-keypass <password>`	The password that protects the private key of the key pair. Press Enter at the prompt to set the key password to the password associated with the keystore. `keypass` must be at least 6 characters long.

Examples

- **Example 1: Display the contents of the keystore –** `keytool -list -keystore <filePath>\keystore.jks -storepass <storepass>`
- **Example 2: Import a certificate reply from a CA –** `keytool -import -file <certificate file> -keystore <filePath>\keystore.jks -`

```
storepass <storepass> -noprompt -trustcacerts -alias
<alias>
```

- **Example 3: Delete a certificate –** `keytool -delete -alias <alias> -keystore <filePath>\keystore.jks -storepass <storepass>`
- **Example 4: Generate a key pair –** `keytool -genkey -keystore <filePath>\keystore.jks`

  The certificate request must be signed by a CA or self-signed by using the `-selfcert` **keytool** option.

Unwired Server Runtime Utilities

Unwired Server runtime supports many utilities used to manage the server environment and its artifacts.

Launch these utilities from the command line, or from Sybase Control Center.

Runtime Configuration (configure-mms) Utility

Applies manual Unwired Server configurations. Use the `configure-mms.bat` utility before restarting Unwired Server. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin`.

Syntax

```
configure-mms [<clusterName>      ]
```

Parameters

- **clusterName –** the name of the cluster to which the Unwired Server belongs.

Usage

In situations that require server configuration changes to be performed outside of Sybase Control Center, use the `configure-mms.bat` utility to commit these changes.

License Upgrade (license) Utility

Upgrades the license in a served model when you purchase more licenses from Sybase. In an unserved model, you simply replace the license file. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin`.

The `license.bat` utility upgrades only Unwired Platform licenses. You cannot use it to upgrade Afaria licenses, which can be upgraded only with the Afaria Administrator.

Syntax

```
license.bat <PE> <LT> [LicenseNumber]
```

Note: `[LicenseNumber]` is required for *<PE> <EE>*. In all other editions, `[LicenseNumber]` is optional.

Parameters

- **PE –** use the corresponding abbreviation that matches the product edition in your license. Valid product editions include:
  - EE for Enterprise Edition
  - ED for Enterprise Developer Edition
  - PD for Personal Developer Edition
- **LT –** use the corresponding abbreviation that matches the license type in your license. Valid license types include:
  - AC for Application deployment CPU license
  - AS for Application deployment seat license
  - CP for CPU license
  - ST for Seat license
  - OT for Other license
  - SS for Standalone seat license
  - DT for Development and test license

Examples

- **Basic Example –** If your license includes an uncommented line that reads `INCREMENT SUP_BASESRVR SYBASE 2010.03316 31-mar-2010 uncounted \ VENDOR_STRING=PE=EE;LT=CP HOSTID=ANY ISSUER="CO=Sybase, \`, use this command.

  ```
  license.bat EE CP <MyLicenseNumber>
  ```

Usage

Depending on the product edition you have, the license type varies according to these guidelines:

- If you entered EE for product edition, the license type can be AC, AS, CP, or ST.
- If you entered PD for product edition, the license type can be SS.
- If you entered ED for product edition, the license type can be DT.

See also

- *Updating and Upgrading Unwired Platform Licenses* on page 186
- *Locating Information in a License File* on page 187

Synchronization Monitor (mlmon) Utility

Monitors the progress of replication synchronization and checks for potential data contention or bottlenecks. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhereXX\BINXX`.

Prerequisites: To use this utility, ensure you are using the correct version of the JDK. For details, see *Unwired Platform Runtime Requirements* in *Supported Hardware and Software* . If you choose to use an existing JDK option during installation, also ensure that you have set the `JAVA_HOME` environment variable to the installation location of this JDK version. Otherwise, you must use a text editor to modify the existing JDK path in the batch file itself.

Syntax

```
mlmon [connect-options|inputfile.{mlm|csv}]
```

When you execute this command, you are prompted to provide:

- Valid administrator credentials
- RBS protocol (default: HTTP)
- RBS port (default: 2480)

Parameters

- **connect-options –** allows you to connect to a messaging server on startup. A monitor connection starts like a synchronization connection to the messaging server. Allowed values are:

Option	Description
`-u ml_username`	Required to connect to the messaging server.
`-p password`	Required to connect to the messaging server.

Option	Description

`-x [tcpip | tls | http | https] [(keyword=value;...)]`	Required to connect to the messaging server. The keyword=value pairs can be:
	• Host – the network name or IP address of the computer where the messaging server is running. By default, it is the computer where the monitor is running.
	• Protocol – should be set to the same network protocol and port as the messaging server is using for synchronization requests.
	• Additional Network Parameters – optional parameters, including:
	  • `buffer_size=number`
	  • `client_port=nnnn`
	  • `client_port=nnnn-mmmmm`
	  • `persistent=[0|1]` (HTTP and HTTPS only)
	  • `proxy_host=proxy_hostname` (HTTP and HTTPS only)
	  • `proxy_port=proxy_portnumber` (HTTP and HTTPS only)
	  • `url_suffix=suffix` (HTTP and HTTPS only)
	  • `version=HTTP-version-number` (HTTP and HTTPS only)

`-o outputfile.{mlm|csv}`	Closes the monitor at the end of the connection and saves the session in the specified file.

- **inputfile.{mlm|csv} –** prompts the monitor to open the specified file.

Package Administration Utilities

Deploy, delete, import, and export application packages using the administration command line utilities. You can execute these commands from the administration command line utility console or from the command line.

To issue commands using the interactive administration command line utility console, open `<<UnwiredPlatform_InstallDir>>\Servers\UnwiredServer\bin\supadmin.bat`. You must enter your host name, port, and administrator credentials before entering a command. Otherwise, the console prompts you for:

- Host – the host name for the Unwired Server. The default value is localhost.
- Port – the port used for the Unwired Server. The default value is 2000.
- Login ID – the administrator login ID to perform authentication with. The default value is supAdmin.

- Password – the administrator password to perform authentication with. The default value is s3pAdmin.

To issue the utility at the command line use:

```
supadmin.bat -host host name -port port -u username -pw password
commandName commandOptions
```

For example, to delete the package `test:1.0` from the domain, enter:

```
supadmin.bat -host test01.sybase.com -port 2000 -u supAdmin -pw PWD
delete -d default -pkg test:1.0
```

Deploy Application Package (supadmin) Utility

Deploys a package to Unwired Server with the administration client APIs.

Syntax

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

Parameters

- **deploy-options –** uses these option to deploy and configure the package:

Option	Description
-d *domain*	Specifies the domain to which the package belongs.
-dcf *descriptorFile*	(Optional) Specifies the descriptor file for the package.

Option	Description
-dm *deploymentMode*	The deployment mode determines how the deployment process handles the objects in a deployment unit and package. Which value you choose depends on whether or not a package of the same name already exists on Unwired Server. Allowed values are: • `UPDATE` – updates the target package with updated objects. After deployment, objects in the server's package with the same name as those being deployed are updated. By default, deploymentMode is UPDATE. • `NOCLOBBER` – deploys the package only if there are no objects in the target server's package that have the same name as any of those objects being deployed. • `REPLACE` – replaces any of the target objects with those in the package. After deployment, the servers package contains only those objects being deployed. • `VERIFY` – do not deploy package. Only return errors, if any. Used to determine the results of the `UPDATE` deploy mode. If the deployment mode is specified both in the descriptor file and the command-line, the command-line deploymentMode option override the deployment mode specified in the descriptor file.
-file *deploymentUnitFileName*	Defines the file name of the deployment unit. For example, `MyDeployUnit.xml`.
-rm *roleMapping*	Defines the role mapping for the package. Accepted values are: • `role1=AUTO` – the logical role defined in the package. • `role2=SUPAdmin,SUPUser` – the physical roles defined in the security provider. The role mapping `-rm role1,role2` maps the logical roles of the package to the physical roles in the server.
-sc *securityConfig*	Defines the security configuration for the package. Select an existing security configuration from the domain to which the package will be deployed.

Option	Description
-sl	Enables silent mode, which disables all user interactive questions during package deployment. During silent mode, the default values for each option are used. This mode is mainly used when writing a batch executing file. For example, the command line `deploy -file deployunit.xml -sl` deploys the package to the default domain with a deploy mode of `UPDATE` and a sync mode of `REPLICATION`, without asking for user confirmation.

Examples

- **Basic Example –** This command updates an existing deployment unit called `samples/uep/deployment/customer_list_unit.xml`. The batch file uses default values for all other command line options:

```
supadmin -host test01.sybase.com -port 2000 -u supAdminID -pw
supPwd deploy -d default -dm update -sc admin -file samples/uep/
deployment/customer_list_unit.xml
-rm "testrole=SUP Admin;testrole1=SUP User,SUP User2"
```

Import Application Package (import) and Export Application Package (export) Utilities

The **import** command imports an existing package to Unwired Server. The **export** command exports a package from Unwired Server.

Syntax

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

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

Parameters

- **import-options | export-options –** use these options to import or export the package:

Option	Usage	Description
-d *domain*	import, export	During import, specifies the domain to which you are importing the package. During export, specifies the domain to which the package currently belongs.
-file *fileName*	import	Specifies the file name of the exported package.
-pkg *packageName*	export	Specifies the name of the package to export.

Examples

- **Import example –** This command imports the "c:/imported.zip" file to the domain "default":

```
supadmin.bat -host test01.sybase.com -port 2000 -u supAdmin -pw
PWD
 import -d default -file c:\imported.zip
```

- **Export example –** This command exports the package "samplePkg" from the domain "default" to make it available to other domains:

```
supadmin.bat -host test01.sybase.com -port 2000 -u supAdmin -pw
PWD
 export -d default -pkg samplePkg -file c:\exported.zip
```

Delete Application Package (delete) Utility

Deletes a package from Unwired Server.

Syntax

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

Parameters

- **delete-options –** use these options to delete the package:

Option	Description
-d *domain*	Specifies the domain to which the package belongs.
-pkg *packageName*	Specifies the name of the package to delete.

Examples

- **–** This command deletes samplePkg from Unwired Server:

```
supadmin.bat -host test01.sybase.com -port 2000 -u supAdmin -pw
PWD delete   -d default -pkg samplePkg
```

Start and Stop sampledb Server (sampledb) Utility

Starts and stops the sampledb server from the command line.

Syntax

```
sampledb.bat [install auto | install manual | start | stop ]
```

Parameters

- **install auto –** installs the sampledb server as a Windows service in auto-start mode.

- **install manual –** installs the sampledb server as a Windows service in manual start mode.
- **start –** starts the sampledb server.
- **stop –** stops the sampledb server.

Advantage Database Server® Backup (adsbackup) Utility

A command line backup utility for use with Advantage Database Server. The **adsbackup** utility collects all the required information to perform a backup or restore, and executes the backup or restore on the specified server. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\Advantage910\Server`.

Prerequisites: The **adsbackup.exe** utility loads `ACE32.DLL` or `libace.so` dynamically so it can be used with different versions of Advantage Client Engine (ACE). The minimum ACE version requirement is v6.0. However, if the server operating system is NetWare, the ACE version must be v7.0 or greater.

Syntax

To back up a directory of free tables:

```
adsbackup.exe [options] <src path> <file mask> <dest path>
```

To back up a data dictionary and its associated tables:

```
adsbackup.exe [options] <src database> <dest path>
```

Parameters

- **src path –** the path to the free tables you want to back up.
- **file mask –** a file mask similar to "*.adt" or "*.dbf" which determines which tables to include in the backup image.
- **dest path –** the backup image destination.
- **src database –** the path to the data dictionary you want to back up.
- **[options] –** these options are available:

Option	Description
-a	Creates a new backup image and initializes the source data so it can be used in a subsequent differential backup.
-c [ANSI|OMI]	Sets the character type of the SQL statement or connection used to open each table and transfer its data. Default: ANSI.
-d	Logs a warning before overwriting existing tables. The default behavior is to overwrite all tables when performing a backup or restore operation.

Option	Description
-f	Performs a differential backup. Includes only tables and records that have been modified since the last backup. Use this option only on databases on which you have previously called **adsbackup -a** to initialize the differential backup.
-h [ON|OFF]	Sets the rights-checking mode (used to open each table and transfer its data) of the SQL statement or connection used by the backup utility.
-i *<file1, file2...>*	Provides a list of tables to include in the backup or restore command. When using a data dictionary, specify the table name in the dictionary. When using free tables, specify the base table name including the file extension; for example, "table1.adt".
-e *<file1, file2...>*	Provides a list of tables to exclude from the backup or restore command. When using a data dictionary, specify the table name in the dictionary. When using free tables, specify the base table name including the file extension; for example, "table1.adt".
-m	Makes a backup or restore copy of only the data dictionary. Therefore, only metadata is backed up.
-n *<path>*	Specifies the location in which to store the backup log table.
-o *<filepath>*	Specifies a file name, along with the file path to the backup log table location.
-p	If the source path is a database, indicates the database password for the user. If the source path is a directory, lists the free table passwords. Free table usage can pass a single password for all encrypted tables, or a name=value pair for each table; for example, "password_for_all" or "table1=pass1;table2=pass2".
-q [PROP|COMPAT]	Specifies the locking mode of the SQL statement or connection used by the backup utility. This is the locking mode used to open each table and transfer its data; either proprietary (PROP) or compatible (COMPAT).
-r	Performs a restore instead of a backup (which is the default behavior).
-s *<server path>*	Specifies the connection path of the ADS server to perform the backup or restore operation. In most situations, **adsbackup** can determine the connection path to use by examining the source path. Use this option if you are having connection problems or want to use a specific connection path when performing the backup or restore.

Option	Description
-t *<server port>*	Defines the server port **adsbackup** connects to when using the Advantage JDBC Driver. This option is valid only with the Java **adsbackup** utility. Default: 6262.
-u [ADT|CDX|NTX]	Sets the table type of the SQL statement or connection used by the backup utility. This is the table type used to open each table and transfer its data. This also determines the table type of the log file produced by the backup or restore procedure.
-v [1-10]	Configures the lowest level of error severity to return. Default: 1.

Usage

You can use **adsbackup** in conjunction with the Unwired Server database file recovery tool **MORecover**. See *Unwired Server Database File Recovery (MORecover) Utility*.

For more information on **adsbackup.exe** usage and sample use cases, see the *Advantage Database Server* documentation.

See also

- *Backing Up Messaging Data* on page 179

Unwired Server Database File Recovery (MORecover) Utility

The **MORecover** utility, with **adsbackup.exe**, backs up Unwired Server database files while the server is running. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\MessagingServer\Bin`.

The Advantage Database Server allows an online backup while the server is running. Online Backup (or "hot" backup) functionality allows you to capture and save a snapshot of your database. If users currently have tables opened and are making changes, the resulting backup image will not include their new modifications, it will only include the data as it existed when the snapshot was taken. See: *http://devzone.advantagedatabase.com/dz/WebHelp/Advantage10.1/index.html?master_online_backup.htm*.

Prerequisite: Before executing **MORecover**:

1. Create an MORecover snapshot by running **adsbackup** to back up the tables required by MORecover. Use the **-i** (include) option to indicate the database tables to include in the backup. For an MORecover snapshot, these tables are:
   - APPLICATIONS
   - CFG_IDS
   - CFG_PROP_VALUES
   - CFG_SUBFOLDER_PROP_VALUES
   - CFG_TEMPLATES

- DEVICES
- USER_DEVICE
- USERS

The source database argument for the command must be the full path of the `OBR.add` file in the Unwired Server data folder. The destination can be any folder. A collection of files constituting the backed-up data is created in the location you specify and stored as the MORecover backup data (see Example 1 below).

2. Convert the files generated by **adsbackup** into a standard database format. The generated files reflect the raw data from the original tables, excluding the index data. Therefore, files generated by **adsbackup** cannot be used directly as a database (see Example 2 below).

For information on **adsbackup** utility parameters and options, see *Advantage Database Server Backup (adsbackup) Utility*.

Syntax

```
MORecover <recoveryDatabaseLocation>
```

Examples

- **Example 1: –** This example uses **adsbackup -i** (include) to indicate the database tables to include in the MORecover snapshot.

  ```
  adsbackup.exe -i
  APPLICATIONS,CFG_IDS,CFG_PROP_VALUES,CFG_SUBFOLDER_PROP_VALUES,CF
  G_TEMPLATES,DEVICES,USER_DEVICE,USERS <sourceDatabaseLocation>
  <destinationFileLocation>
  ```

- **Example 2: –** Before running MORecover to restore your database, you must convert the data files created by **adsbackup** back to a standard database format. The following example uses the **-r** option to restore the backup data to a temporary folder. You can then point MORecover to the restored data in this folder.

  ```
  adsbackup.exe -r <destinationFileLocation>
  <recoveryDatabaseLocation>
  ```

Usage

While the specified subset of tables typically results in a time-efficient backup operation, Sybase recommends that you schedule this command to run regularly during a time of light system load, since the backup operation contends with Unwired Server normal processing for database resources.

See also

- *Restoration of the Messaging Data* on page 181

Update Properties (updateprops.bat) Utility

Performs multiple functions, including registering or removing a participating node for a cluster, or update a specific server property.

Note: Unless documented otherwise, use Sybase Control Center to change most properties in the runtime to avoid unnecessary complication.

Syntax

```
updateprops.bat [-u username] [-p password] [-d dsn] [-f
propertyFile]
[-cn clusterName] [-nv "<propertyName=NewValue>"] [-r] [-v] [-x]
```

Parameters

- **-u *username* –** the platform administrator username.
- **-p *password* –** the platform administrator password.
- **-d *dsn* –** the data source name (DSN) of the cluster database.
- **-cn *clusterName* –** the name that identifies the Unwired Platform Cluster
- **-nv "<*propertyName=NewValue*>" –** one or more platform property values that requires change. Multiple values can be defined; however, they must be separated by the pound symbol (#). For example:

  ```
  –nv
  “ml.threadcount=10#sup.admin.port=2005#sup.sync.port=2490"
  ```
- **-v –** use verbose output in the command window.

Examples

- **Changing a cdb threadcount property –** Update the ml.threadcount property of the production environment cache database to `20` by running:

  updateProps.bat -nv "ml.threadcount=20"

  This is only recommended for deployment editions of Unwired Platform.
- **Printing out cluster information –** Use the utility command without options:

  updateProps.bat -nv "ml.threadcount=20"

  This command displays current ThreadMonitor, DataSource for culsterdb and other information. For example:

  ```
  0 [main] INFO com.sybase.djc.util.ThreadMonitor  - init:
  minimumActiveThreads =
  0 (unlimited)
  3 [main] INFO com.sybase.djc.util.ThreadMonitor  - init:
  maximumActiveThreads =
  0 (unlimited)
  ```

```
3 [main] INFO com.sybase.djc.util.ThreadMonitor  - init:
initialThreadLimit = 0
(unlimited)
4 [main] INFO com.sybase.djc.util.ThreadMonitor  - init:
targetResponseTime = 0
(unchecked)
11 [main] INFO com.sybase.djc.sql.DataSource  - dataSourceClass =
class com.syba
se.jdbc3.jdbc.SybDataSource
23 [main] INFO com.sybase.djc.sql.DataSource  - ant.project =
default-data-sourc
es
24 [main] INFO com.sybase.djc.sql.DataSource  - dataSourceClass =
com.sybase.jdb
c3.jdbc.SybDataSource
25 [main] INFO com.sybase.djc.sql.DataSource  - databaseType =
Sybase_ASA
25 [main] INFO com.sybase.djc.sql.DataSource  - maxPoolSize = 100
26 [main] INFO com.sybase.djc.sql.DataSource  - password = ******
26 [main] INFO com.sybase.djc.sql.DataSource  - portNumber = 5200
27 [main] INFO com.sybase.djc.sql.DataSource  - serverName =
localhost
27 [main] INFO com.sybase.djc.sql.DataSource  - serviceName =
clusterdb
28 [main] INFO com.sybase.djc.sql.DataSource  - user = dba
28 [main] INFO com.sybase.djc.sql.DataSource  -
jdbc:DISABLE_UNPROCESSED_PARAM_W
ARNINGS = true
29 [main] INFO com.sybase.djc.sql.DataSource  -
jdbc:IS_CLOSED_TEST = INTERNAL
30 [main] INFO com.sybase.djc.sql.DataSource  - jdbc:SERVICENAME =
clusterdb
30 [main] INFO com.sybase.djc.sql.DataSource  -
jdbc:networkProtocol = Tds
31 [main] INFO com.sybase.djc.sql.DataSource  - jdbc:portNumber =
5200
31 [main] INFO com.sybase.djc.sql.DataSource  - jdbc:serverName =
localhost
1000 [main] INFO org.mortbay.log  - Logging to
org.slf4j.impl.Log4jLoggerAdapter
(org.mortbay.log) via org.mortbay.log.Slf4jLog
1001 [main] INFO org.mortbay.log  - ClusterRegistration completed
successfully.
```

Usage

Before running this utility, ensure that the data tier is available; otherwise platform data is not modified correctly.

See also

- *Changing the Cache Database Server Thread Count* on page 21
- *Changing Database Ports for SQLAnywhere Databases* on page 19
- *Changing SQLAnywhere Database Server Startup Options* on page 20

DOE-C Utilities

To perform tasks needed when working with Sybase SAP DOE Connector.

esdma-converter Command

Converts an SAP ESDMA bundle resource metadata file to an Unwired Platform package.

The **esdma-converter** executable file is located in the *<UnwiredPlatform_InstallDir>*\UnwiredPlatform\MobileSDK\DOE-C_SDK\bin directory.

Syntax

```
esdma-converter esdma-bundle-dir [-afx afx_file]
[-dsd output_file] [-esdma bundle_metadata_file] [-help]
```

Parameters

- ***esdma-bundle-dir* –** the source directory for the ESDMA resource metadata file to be converted.
- **-afx –** the AFX (application from XML) output document file. The default is *esdma-bundle-dir*/META-INF/afx-esdma.xml.
- **-dsd –** the DOE-C output document name. The default is *esdma-bundle-dir*/META-INF/ds-doe.xml.
- **-esdma –** the source ESDMA bundle resource metadata file. The default is *esdma-bundle-dir*/Resources/AnnotatedMeta.xml, or *esdma-bundle-dir*/Resources/Meta.xml if the former is not found.
- **-help –** gets help on this command.

Code Generation Utility Command Line Reference

Generates platform-specific code for a message-based application from an ESDMA bundle.

Before using the Code Generation Utility:

- Copy your ESDMA .zip file into an ESDMA directory (*<ESDMA_dir>*) and extract its contents into that directory.
- Verify that the following files are present in the META-INF directory under your *<ESDMA_dir>* directory.
  - sup-db.xml
  - afx-esdma.xml
  - ds-doe.xml

Note: *<ESDMA_dir>*\META-INF\sup-db.xml is also referred to as the AFX document.

codegen Command

Command syntax and parameter descriptions.

Run `codegen.bat` to execute the Code Generation Utility. The `codegen.bat` file is located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\MobileSDK\DOE-C_SDK\bin`.

Syntax

```
codegen -android|-cs|-oc|-rim -client -doe -sqlite|-ulj -log:co
[-output <output_dir>] [-doc] <ESDMA_dir>\META-INF\sup-db.xml
```

All parameters not enclosed by "[...]" are required for use with DOE-C.

Parameters

- **-cs –** for Windows and Windows Mobile, generates C# source files.
- **-android –** for Android, generates Java source files.
- **-oc –** for iPhone, generates Object C source files.
- **-rim –** for BlackBerry, generates Java source files.
- **-client –** generates client side source code.
- **-doe –** generates code for a DOE-based synchronization package.
- **-sqlite –** for Windows and Windows Mobile, and iPhone, generates code for database type SQLite.
- **-ulj –** for BlackBerry, generates code for database type UltraLiteJ.
- **-log:co –** generates a client-only log record.
- **-output –** specifies the output directory where generated code is placed. Defaults to `<UnwiredPlatform_InstallDir>UnwiredPlatform\ClientAPI\utils\genfiles`.
- **-doc –** generates documentation for the generated code.

SAP DOE Connector Command Line Utility

A text-based console that allows you to manage ESDMA packages and subscriptions to those packages without going through Sybase Control Center.

Work interactively in the Command Line Utility console until you become familiar with the command options. The console prompts you for all required parameters and lets you choose values from a list, when there is a fixed list of valid values; for example, for domain name.

Write batch files to silently execute any sequence of commands.

You must use the Command Line Utility's **deploy** command to deploy an ESDMA package. The functionality of all the other commands is available through Sybase Control Center. See *Sybase Control Center > Deploy > Packages*.

Note: Before you attempt to deploy a package, set up the security configuration in Sybase Control Center for that package to use. See Sybase Control Center > Administer > Security Configurations.

Table 26. Command Line Utility Notation Conventions

Symbols	Meaning	Example
[...]	Optional – parameters not enclosed by this symbol are required	`[-h|--help]` Getting help is an option on every command.
|	Alternatives – use one or the other, not both	`[-h|--help]` You can enter the help parameter as `-h` or as `--help`.
{...}	Grouping – alternatives symbol applies to all parameters enclosed	`{-u|--technicalUser` *`SAPUserAccount`* `-pw|--password` *`SAPUserPassword`*`} | {-ca|-certAlias` *`certificateAlias`*`}` You must enter either the technical user ID and password, or the certificate alias.

Starting the Command Line Utility Console

Before you can use the DOE-C Command Line Utility interactively, you must start the console.

1. In Windows Explorer or at a command prompt, navigate to `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\doe-c_clu\bin`.
2. Start up `clu.bat`.
3. Log in, or enter commands without a login.

   If you enter a command (other than **help** or **exit**) without first logging in, you must enter the DOE-C server admin listener URL, user name, and password when you are prompted for the first command that you enter. You are not prompted for this information again when you enter additional commands.

Running Commands in Batch Mode

In batch mode, the DOE-C Command Line Utility takes commands from an XML file instead of requiring you to enter them interactively through the console.

Creating an XML File to Run Commands in Batch Mode

To run commands in batch mode, you must enter them into an XML file with special tagging.

To make your batch file run smoothly:

- Use the silent option with each command. See *Using the Silent Option* on page 256.
- Specify all required parameters for each command.

1. In a text editor, create a file with the XML extension.
2. Open the file and enter these first two lines:
```
<?xml version="1.0" encoding="UTF-8"?>
<commands>
```
3. Enter these lines for the **login** command:
```
    <command name="login" sequence="1">
        <option name="url" arg="DOECSocketListenerUrl" />
        <option name="pw" arg="DOECUserPassword" />
        <option name="u" arg="DOECUuser" />
    </command>
```
4. For each command you want to execute, enter the information in an XML structure similar to this:
```
    <command name="commandName" sequence="sequenceInteger">
        <option name="optionNamel" arg="optionValue1" />
        <option name="optionName2" arg="optionValue2" />
        <option name="optionName3" arg="optionValue3" />
            ...
        <option name="optionNameN" arg="optionValueN" />
    </command>
```
   The sequence parameter controls the order in which the commands are executed.
5. After the last <command> entry, terminate the file:
```
</commands>
```

Running the Command Line Utility in Batch Mode

The execute-commandXMLFile command runs the Command Line Utility in batch mode.

Prerequisites

Create an XML file that contains the commands that you want to execute, with proper XML tagging.

Task

1. At a command prompt, navigate to `%DOE-C_CLU_HOME%/bin/`.
2. Enter:
```
execute-commandXMLFile xmlFileName
```
   where *xmlFileName* is either the full path to the file containing the commands to be executed, or the relative path to that file from the `%DOE-C_CLU_HOME%/bin/` directory.

Using the Silent Option

Most of the commands in the DOE-C Command Line Utility support the silent option, which suppresses all user prompts, and which, in general, you want to do for batch execution.

Before using the silent option (**-sl|--silent**), verify that suppressing user prompts does not have undesirable results.

Here are some examples that illustrate potentially undesirable results:

- `getPackages -o pac.xml -sl` – if `pac.xml` already exists, it is overwritten without confirmation.
- `setPackageLogLevel -l DEBUG -i pac.xml` – if `pac.xml` contains more than one package, the log level for all packages is set to DEBUG.

Command Summary

A summary of DOE-C Command Line Utility commands. For more detailed information about a command, refer to the reference topic for the command.

Table 27. Administrative commands

Operation	Command
Start Command Line Utility console to enter commands interactively	`<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\doe-c_clu\bin\CLU.bat` (from a command prompt)
Run Command Line Utility to take commands from an XML file	`<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\doe-c_clu\bin\executeCommandXMLFile.bat xmlFileName` (from a command prompt)
Log in	`login -u|--DOEServerUser UnwiredServerUser -pw|--password UnwiredServerUserPassword -url|--DOECSocketListenerUrl Url [-h|--help] [-sl|--silent]`
Exit	`exit [-h|--help] [-sl|--silent]`
Get help	`help [commandName | [-a|--all]] | [-h|--help]`

Table 28. Package management commands

Operation	Command
Deploy a package	`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]`
Get details for specific deployed packages	`getPackages {-i|--in inputXmlFile} | {-d|--domain domainName [-ps|--packageNames nameAndVersionList]} [-o|--out outputXmlFile] [-h|--help] [-sl|--silent]` If you omit **-ps**, **getPackages** returns a list of all deployed packages
Set endpoint properties for a deployed package	`setEndpointProperties {-i|--in inputXmlFile} | {-d|--domain domainName {-a|--all | -ps|--packageNames nameAndVersionList} {-u|--technicalUser SAPUserAccount -pw|--password SAPUserPassword} | {-ca|-certAlias certificateAlias} [-ds|--doePacketDropSize byteSize] [-ew|--doeExtractWindow maxNumMsgs] [-t|--soapTimeout seconds] [-h|--help] [-sl|--silent]`
Get endpoint properties for a deployed package	`getEndpointProperties {-i|--in inputXmlFile} | {-d|--domain -a|--all | -ps|--packageNames name} [-h|--help] [-sl|--silent]`
Test endpoint properties for a deployed package	`testEndpoint {-i|--in inputXmlFile | {-d|--domain domainName -p|--packageName name}} [-h|--help] [-sl|--silent]`
Set the security configuration for deployed packages	`setPackageSecurityConfiguration {-i|--in inputXmlFile} | {-d|--domain domainName {-a|--all | -ps|--packageNames nameAndVersionList}} [-sc|--securityConfiguration securityConfigName [-h|--help] [-sl|--silent]`

Operation	Command
Set the log level for deployed packages	setPackageLogLevel {-i|--in *inputXmlFile*} | {-d|--domain *domainName* {-a|--all | -ps|--packageNames *nameAndVersionList*}} [-l|--logLevel *level* [-h|--help] [-sl|--silent]
Get the log level for deployed packages	getPackageLogLevel {-i|--in *inputXmlFile*} | {-d|--domain *domainName* {-a|--all | -ps|--packageNames *nameAndVersionList*}} [-h|--help] [-sl|--silent]
Remove deployed packages	removePackages {-i|--in *inputXmlFile*} | {-d|--domain *domainName* -ps|--packageNames *name*} [-h|--help] [-sl|--silent]

Table 29. Subscription management commands

Operation	Command
Get information on subscriptions to deployed packages	getSubscriptions {-i|--in *inputXmlFile*} | {-d|--domain *domainName* -ps|--packageNames *nameAndVersionList*} [-o|--out *outputXmlFile*] [-f|--filter *filterExpression*] [-h|--help] [-sl|--silent]
Get information on subscriptions to a deployed package, with sorting and pagination options	getSubscriptions2 {-i|--in *inputXmlFile*} | {-d|--domain *domainName* -p|--packageName *name*} [-o|--out *outputXmlFile*] [-f|--filter *filterExpression*] [-pn|--pageNumber *number*] [-ps|--pageSize *size*] [-s|--sort *column*[:Ascending|Descending]] [-h|--help] [-sl|--silent]
Set the log level for subscriptions to a deployed package	setSubscriptionsLogLevel {-i|--in *inputXmlFile*} | {-d|--domain *domainName* -p|--packageName *name* {-a|--all | -s|--subscriptionID *ID*}} -l|--logLevel *level* [-h|--help] [-sl|--silent]

Operation	Command
Get the log level for subscriptions to a deployed package	`getSubscriptionsLogLevel {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name -s|--subscriptionID ID} [-h|--help] [-sl|--silent]`
Suspend subscriptions	`suspendSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} [-h|--help] [-sl|--silent]`
Resume subscriptions	`resumeSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} [-h|--help] [-sl|--silent]`
Resynchronize subscriptions to a deployed package	`resyncSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} [-h|--help] [-sl|--silent]`
End subscriptions to a deployed package	`endSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} [-h|--help] [-sl|--silent]`

Console Management Commands

Use the administrative commands to start the Command Line Utility console, log in, get help, and exit.

login Command

Logs in to the DOE-C Command Line Utility console.

If you do not use the **login** command to log in to the Command Line Utility console, you are prompted to enter the login information for the first command (other than **help** or **exit**) that you enter.

Syntax

```
login -u|--DOEServerUser UnwiredServerUser
-pw|--password UnwiredServerUserPassword
```

```
-url|--DOECSocketListenerUrl Url
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-u|--DOEServerUser –** specifies the Unwired Server admin user account.
- **-pw|--password –** specifies the Sybase Control Center admin user account password.
- **-url|--DOECSocketListenerUrl –** specifies the URL for the Unwired Server IIOP administration port; this is the same port specified by the `sup.admin.port` attribute in the `sup.properties` file. This port is set during installation of Sybase Unwired Platform.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

exit Command

Closes the Command Line Utility console.

Syntax

```
exit [-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

help Command

Displays help text for any specific DOE-C command, or for all commands.

Syntax

```
help [commandName | [-a|--all]] | [-h|--help]
```

Parameters

- **-h|--help –** gets help on the **help** command.
- **-a|--all –** gets help on all commands.
- ***commandName* –** gets help on the specified command.

Aborting Commands

There are two ways to abort commands in interactive mode.

Abort method	Description
Press Ctrl+C at any time	Aborts in-progress command and exits from Command Line Utility console. • Works at any time with any command. • To enter additional commands after using this option, you must restart the console.
Enter "abort" when prompted	Aborts in-progress command without exiting from Command Line Utility console. • Works only with certain commands, and only when prompted. • After using this option, you can continue entering commands.

Package Management Commands

Manage DOE-C packages from the Command Line Utility, rather than from Sybase Control Center.

deploy Command

Deploys a DOE-C package to Sybase Unwired Server.

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.

Syntax

```
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]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-a|--applicationID –** specifies the application ID.

- **-dir|--deployFilesDirectory –** specifies the directory location that contains deployment files.
- **-sc|--securityConfiguration –** specifies the name of the security configuration to use, as defined in Sybase Control Center.
- **-u|--technicalUser –** specifies the SAP technical user account to use when sending non-client-based requests. If `technicalUser` is specified, you must also specify `password`, and you must not specify `certAlias`.
- **-pw|--password –** specifies the SAP technical user account password. Required if `technicalUser` is specified.
- **-ca|--certAlias –** specifies the certificate alias. If `certAlias` is specified, you must not specify `technicalUser`.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

See also

- *Deploying a DOE Package* on page 85

getPackages Command

Generates a list of deployed DOE-C packages, or returns detailed information for one or more specified packages.

Syntax

```
getPackages
{-i|--in inputXmlFile} |
{-d|--domain domainName

[-ps|--packageNames nameAndVersionList]}
[-o|--out outputXmlFile]
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. Generate the XML file using the **-o** parameter.
- **-o|--out –** saves command output to an XML file.
- **-ps|--packageNames –** specifies one or more package names for which detailed information is returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -ps myPkg1:2.0,myPkg2:1.0
  ```

Note: If you omit this parameter, **getPackages** returns a list of all deployed packages.

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

setEndpointProperties Command

Sets the DOE endpoint properties for deployed DOE-C packages.

You must set some of the DOE endpoint properties so that DOE-C can communicate with the SAP server. You can set other DOE endpoint properties to tune the performance of the communications.

Syntax

```
setEndpointProperties
{-i|--in inputXmlFile} |
{-d|--domain domainName
{-a|--all | -ps|--packageNames nameAndVersionList}
{-u|--technicalUser SAPUserAccount
-pw|--password SAPUserPassword} |
{-ca|-certAlias certificateAlias}
[-ds|--doePacketDropSize byteSize]
[-ew|--doeExtractWindow maxNumMsgs]
[-t|--soapTimeout seconds]
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-a|--all –** sets endpoint properties for all deployed packages.
- **-ps|--packageNames –** specifies one or more package names for which endpoint properties are returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -ps myPkg1:2.0,myPkg2:1.0
  ```

- **-ds|--doePacketDropSize –** the size, in bytes, of the largest JavaScript Object Notation (JSON) message that the DOE connector processes on behalf of a JSON client. The default is 1048576 bytes (1MB). The packet drop threshold size should be carefully chosen, so that it is larger than the largest message sent from the DOE to the client, but smaller than the maximum message size which may be processed by the client. Messages larger than the packet drop threshold size cause the subscription to enter the DOE packet drop state and become unusable.

  Note: Do not set lower than 4096.

- **-ew|--doeExtractWindow –** specifies the number of messages allowed in the DOE extract window. When the number of messages in the DOE extract window reaches 50% of this value, DOE-C sends a `StatusReqFromClient` message to advise the SAP DOE system of the client's messaging status and acknowledge the server's state. The default value is 50.
- **-u|--technicalUser –** specifies the SAP technical user account to use when sending non-client-based requests.
- **-pw|--password –** specifies the SAP technical user account password.
- **-ca|--certAlias –** specifies the certificate alias. If `certAlias` is specified, you must not specify `technicalUser`.
- **-t|--soapTimeout –** specifies the DOE SOAP timeout value, in seconds, to use when sending messages to the SAP DOE.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getEndpointProperties Command

Gets the DOE endpoint properties (**HTTPTimeout** value) for a deployed DOE-C package.

Syntax

```
getEndpointProperties
{-i|--in inputXmlFile} |
{-d|--domain -a|--all | -ps|--packageNames name}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads command input from an XML file.
- **-a|--all –** returns endpoint properties for all deployed packages.
- **-ps|--packageNames –** specifies one or more package names for which endpoint properties are returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -ps myPkg1:2.0,myPkg2:1.0
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

testEndpoint Command

Tests the DOE endpoint accessibility for a deployed DOE-C package.

Use the **testEndpoint** command to verify that the DOE endpoint is accessible using the parameters you set with the **setEndpointProperties** command.

Syntax

```
testEndpoint
{-i|--in inputXmlFile | {-d|--domain domainName
-p|--packageName name}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-p|--packageName –** specifies package name for which endpoint properties are set. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

setPackageSecurityConfiguration Command

Sets the security configuration for a deployed DOE-C package.

Syntax

```
setPackageSecurityConfiguration
{-i|--in inputXmlFile} |
{-d|--domain domainName
{-a|--all | -ps|--packageNames nameAndVersionList}}
[-sc|--securityConfiguration securityConfigName
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-sc|--securityConfiguration –** specifies the security configuration name, defined in Sybase Control Center, to use with the specified packages.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-a|--all –** sets the security configuration for all deployed packages.
- **-ps|--packageNames –** specifies one or more package names for which the security configuration is set. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -ps myPkg1:2.0,myPkg2:1.0
  ```

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

setPackageLogLevel Command

Sets the log level, which determines the amount of information logged, for one or more deployed DOE-C packages.

Syntax

```
setPackageLogLevel
{-i|--in inputXmlFile} |
{-d|--domain domainName
{-a|--all | -ps|--packageNames nameAndVersionList}}
[-l|--logLevel level
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-l|--logLevel –** specifies the log level to be set:
  - **OFF** – no information is logged.
  - **ERROR** – only error messages are logged.
  - **WARN** – adds less serious warnings to information logged by ERROR.
  - **INFO** – adds informational messages to information logged by WARN.
  - **DEBUG** – provides the maximum amount of detail that can be logged.
- **-a|--all –** sets the log level for all deployed packages.
- **-ps|--packageNames –** specifies one or more package names for which the log level is set. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getPackageLogLevel Command

Gets the log level for one or more deployed DOE-C packages.

Syntax

```
getPackageLogLevel
{-i|--in inputXmlFile} |
{-d|--domain domainName
{-a|--all | -ps|--packageNames nameAndVersionList}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the -o parameter with the getPackages command.
- **-ps|--packageNames –** specifies one or more package names for which detailed information is returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -p myPkg:2.0
  ```

  Note: If you omit **-ps**, **getPackageLogLevel** returns a list of log levels for all deployed packages.
- **-h|--help –** gets help on this command.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

removePackages Command

Removes one or more deployed DOE-C packages from the Sybase Unwired Server.

Syntax

```
removePackages
{-i|--in inputXmlFile} |
{-d|--domain domainName -ps|--packageNames name}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the -o parameter with the getPackages command.
- **-ps|--packageNames –** specifies one or more package names to be removed. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -p myPkg:2.0
  ```

  Note: If you omit **-ps**, **removePackages** prompts interactively for package names to be removed.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

Subscription Management Commands

Manage DOE-C package subscriptions from the Command Line Utility, rather than from Sybase Control Center.

getSubscriptions Command

Gets information on subscriptions to one or more deployed DOE-C packages.

Syntax

```
getSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-ps|--packageNames nameAndVersionList}
[-o|--out outputXmlFile]
[-f|--filter filterExpression]
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-o|--out –** saves command output to an XML file.
- **-f|--filter –** specifies the filter to use on the subscriptions. Each column name is followed by a colon and the filter string. Use a comma to separate the information for multiple column names, with no white space; for example:

  ```
  -f columnName:filterString,columcName2:filterString2
  ```

  Valid filter column names are: subscriptionID, packageName, clientID, physicalID, logicalID, userName, language, clientMsgID, clientMsgTimeStamp, serverMsgID, serverMsgTimeStamp, logLevel.

  You can use "?" and "*" wildcard characters in your filter strings; for example:

  ```
  -f clientMsgTimeStamp:*Jan*21?41*2009,userName:john*
  ```

- **-ps|--packageNames –** specifies one or more package names for which subscription information is returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -p myPkg:2.0
  ```

  Note: If **-ps** is omitted, **getSubscriptions** prompts you for a package name.

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getSubscriptions2 Command

Gets information on subscriptions to a deployed DOE-C packages, with output paginated and sorted.

Syntax

```
getSubscriptions2
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name}
[-o|--out outputXmlFile]
[-f|--filter filterExpression]
[-pn|--pageNumber number] [-ps|--pageSize size]
[-s|--sort column[:Ascending|Descending]]
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-o|--out –** saves command output to an XML file.
- **-f|--filter –** specifies the filter to use on the subscriptions. The filter expression must have one column name, followed by a colon and the filter string; for example:

  ```
  -f columnName:filterString
  ```

  Valid filter column names are: subscriptionID, packageName, clientID, physicalID, logicalID, userName, language, clientMsgID, clientMsgTimeStamp, serverMsgID, serverMsgTimeStamp, logLevel.

  You can use "?" and "*" wildcard characters in your filter strings; for example:

  ```
  -f clientMsgTimeStamp:*Jan*21?41*2009
  ```
- **-p|--packageName –** specifies package name for which subscription information is returned. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-ps|--pageSize –** specifies the page size, which is the number of subscriptions per page returned. Page size must be 1 or higher. If you do not specify a page size:
  - If the number of subscriptions returned is greater than 10, you are prompted to enter a page size.
  - If the number of subscriptions returned is 10 or fewer, all subscriptions are listed on one page.

- **-pn|--pageNumber –** specifies the page number, which is the number of the page returned, determined by the page size. Page number must be 1 or higher. If you do not specify a page number:
  - If the number of subscriptions returned is greater than the page size, you are prompted to enter a page number.
  - If the number of subscriptions returned is not greater than the page size, all subscriptions are listed on one page.

  Page size and page number together determine the subscriptions actually returned by for the specified package name; for example, you might specify a page size of 3 with a page number of 2:

  ```
  getSubscriptions2 -p myPkg:2.0 -ps 3 -pn 2
  ```

  This example returns the second page of subscriptions for version 2.0 of the package named myPkg. That page would contain subscriptions 4-6 to the package. With a page size of 3, the fist page would contain subscriptions 1-3, the third page would contain subscriptions 7-9, and so on. If sorting or filtering are specified, these operations produce the list of subscriptions to which page size and page number are applied.
- **-s|--sort –** specifies the columns on which output is to be sorted. If you specify only a column name, the default sort order is ascending; for example:

  ```
  -s UserName
  ```

  Add a colon, followed by `Descending` after the column name to sort in descending order; for example:

  ```
  -s ServerMsgTimeStamp:Descending
  ```

  Valid sort column names are: ClientID, PhysicalID, SubscriptionID, LogicalID, PushQueue, UserName, Language, LogLevel, ServerMsgID, ServerMsgTimeStamp, ClientMsgID, ClientMsgTimeStamp, ApplicationName, and MMSPID.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

setSubscriptionsLogLevel Command

Sets the log level, which determines the amount of information logged, for subscriptions to a deployed DOE-C package.

Syntax

```
setSubscriptionsLogLevel
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name {-a|--all |
-s|--subscriptionID ID}}
-l|--logLevel level
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the -o parameter with the getPackages command.
- **-l|--logLevel –** specifies the log level to be set:
  - **OFF** – no information is logged.
  - **ERROR** – only error messages are logged.
  - **WARN** – adds less serious warnings to information logged by ERROR.
  - **INFO** – adds informational messages to information logged by WARN.
  - **DEBUG** – provides the maximum amount of detail that can be logged.
- **-a|--all –** specifies the log level for all deployed packages.
- **-p|--packageName –** specifies a package name for which the log level is set. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs for which you want to set the log level. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getSubscriptionsLogLevel Command

Gets the log level for subscriptions to a deployed DOE-C package.

Syntax

```
getSubscriptionsLogLevel
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
-s|--subscriptionID ID}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads command input from an XML file.
- **-p|--packageName –** specifies a package name for which detailed information is returned. Package name is followed by a colon and the package version number, with no white space; for example:

```
-p myPkg:2.0
```

- **-s|--subscriptionID –** specifies one or more subscription IDs for which you want to get the log level. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```

  Note: If **-s** is omitted, **getSubscriptionsLogLevel** returns a list of all subscriptions for the specified package.

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

suspendSubscriptions Command

Use the **suspendSubscriptions** command to suspend subscriptions to one or all deployed DOE-C packages.

Syntax

```
suspendSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
{-a|--all | -s|--subscriptionID ID}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads subscription ID and package name from input XML file. You can generate the XML file by using the **-o** parameter with the **getSubscriptions** command.
- **-a|--all –** suspends subscriptions for all deployed packages.
- **-p|--packageName –** specifies a package name for which subscriptions are suspended. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```

- **-s|--subscriptionID –** specifies one or more subscription IDs which you want to suspend. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```

- **-sl|--silent –** disables all user interactive questions; this option is mainly used when writing a batch file.

resumeSubscriptions Command

Use the **resumeSubscriptions** command to resume subscriptions to one or all deployed DOE-C packages for which subscriptions have been suspended.

Syntax

```
resumeSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
{-a|--all | -s|--subscriptionID ID}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads subscription ID and package name from input XML file. You can generate the XML file by using the **-o** parameter with the **getSubscriptions** command.
- **-a|--all –** resumes subscriptions for all deployed packages.
- **-p|--packageName –** specifies a package name for which subscriptions are resumeed. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs which you want to resume. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```
- **-sl|--silent –** disables all user interactive questions; this option is mainly used when writing a batch file.

resyncSubscriptions Command

Unblocks DOE queues.

If the Sybase SAP DOE Connector does not respond to the SAP DOE quickly enough, the DOE may mark that subscription's queues as "blocked" and stop sending messages to the DOE-C. At start-up, the DOE-C sends a RestartQ message to the DOE that should unblock these queues. If this happens at times other than at start-up, you can use **resyncSubscriptions** to resume communication from the DOE to the DOE-C.

Syntax

```
resyncSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
{-a|--all | -s|--subscriptionID ID}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.

- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with the **getPackages** command.
- **-a|--all –** reactivates subscriptions for all deployed packages.
- **-p|--packageName –** specifies package name for which subscriptions are reactivated. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs to recover. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

endSubscriptions Command

Ends subscriptions to a deployed DOE-C package.

Syntax

```
endSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
{-a|--all | -s|--subscriptionID ID}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with the **getPackages** command.
- **-a|--all –** ends subscriptions for all deployed packages.
- **-p|--packageName –** specifies package name for which subscriptions are ended. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs to recover. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

Configuration Files

Configuration files hold the initial settings for various Unwired Platform components or subcomponents.

All Unwired Platform components read their configuration files at start-up. Some components check configuration files for changes periodically. Administrators can instruct Unwired Server to reread configuration files, and apply changes to the current process. However, some configuration changes require you to restart the Unwired Server.

Unwired Server Configuration Files

Use Unwired Server configuration files to manually modify server security, logging, and subcomponent configurations.

Sybase recommends that you edit these `.properties` and `.xml` files only if you cannot use Sybase Control Center to configure Unwired Server properties.

See also

- *Sybase Control Center Configuration Files* on page 288
- *Relay Server Configuration Files* on page 291

Global Unwired Server Properties (sup.properties) Configuration File Reference

`sup.properties` is a global properties file that allows you to configure many subcomponents of the Unwired Server. This file is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer`.

Property	Default	Description
sup.admin.port	2000	The port used by the embedded application server.
sup.admin.protocol	IIOPS	The protocol use for Unwired Server administration from Sybase Control Center. Accepted values are IIOP (unsecure) or IIOPS (encrypted).
sup.admin.httpports	8000	The HTTP ports used by the embedded application server.
sup.admin.httpsports	8001, 8002	The HTTPS (secure) ports used by the embedded application server.

Property	Default	Description
sup.admin.iiopsport	2001	The IIOPS port used by Sybase Control Center.
sup.sync.sslkeystore	`Repository/Security/keystore.jks`	The relative path to the Unwired Server keystore file.
sup.sync.sslkeystore_password	changeit	The password to unlock the keystore file
sup.sync.ssltruststore	`Repository/security/truststore.jks`	The relative path to the Unwired Server truststore file.
sup.sync.ssltruststore_password	changeit	The password to unlock the truststore file
sup.sync.port	2480	The synchronization port.
sup.sync.httpsport	2481	The secure synchronization port.
sup.sync.protocol	HTTP, HTTPS	The protocol used for synchronization. By default HTTP; however, you can also use HTTPS.
sup.sync.certificate	`Repository/Certificate/https_server_identity.crt`	The fully qualified path to the certificate file. The value should be the relative path to Unwired Server.
sup.sync.certificate_password	changeit	The password to unlock the certificate.
sup.cluster.name	<*computerName*>	The Unwired Server cluster name.
sup.user.options	n/a	The Unwired Server user startup options.
sup.install.number	<number>	The Unwired Platform install number.
sup.java.install	n/a	The Unwired Platform Java install number.
sup.host	<computerName>	The host name of the Unwired Server.
sup.install.asservice	false	Indicates whether or not Unwired Server is installed in service mode.
sup.node.status	resumed	The Unwired Server cluster status: suspend, resume, pending, suspended, resumed, pending.
sup.node.home	n/a	Messaging service home path.

Property	Default	Description
sup.imo.upa	n/a	Encrypted user name and password of admin@system.
sup.msg.outbound_queue_prefix	sup.mbs.moca.	The outbound queue prefix.
sup.msg.inbound_queue_prefix	sup.mbs.	The inbound queue prefix.
sup.msg.inbound_count	25	The number of inbound queues.
sup.msg.outbound_count	5	The number of outbound queues.
ml.threadcount	5	The synchronization threadcount. This value must be 5 units lower than sqlany.threadcount.
ml.servername	none	The Unwired Server name.
ml.cachesize	50M	The maximum size for the replication protocol server memory cache.
relayserver.trusted_certs	none	The trusted certificate path relative to Unwired Server home.
cdb.threadcount	20	The maximum number of tasks that the cache database server can execute concurrently.
cdb.databasename	default	The cache database name.
cdb.password	sql	The password for the cache database.
cdb.asa.mode	primary	The database mode when cache database type is SQL Anywhere (Sybase_ASA).
cdb.serverport	5200	The cache database port number.
cdb.dsnname	default-cdb	The cache database DSN name.
cdb.install_type	default	The cache database install type.
cdb.username	dba	The cache database user name.
cdb.type	Sybase_ASA	The cache database type.
cdb.serverhost	localhost	The cache database server host name.
cdb.user.options	none	The cache database user-specified startup options.
cdb.servername	none	The cache database server name.
cldb.serverhost	localhost	The cluster database server host name.

Property	Default	Description
cldb.serverport	5200	The cluster database port number.
cldb.username	dba	The cluster database user name.
cldb.password	sql	The cluster database password.
cldb.databasename	clusterdb	The cluster database name.
cldb.dsnname	none	The cluster database DSN name.
cldb.type	Sybase_ASA	The cluster database type.
monitoringdb.server-host	localhost	The monitoring database server host name.
monitoringdb.server-port	5200	The monitoring database port number.
monitoringdb.user-name	dba	The monitoring database user name.
monitoringdb.pass-word	sql	The monitoring database password.
monitoringdb.databa-sename	monitordb	The monitoring database name.
monitoringdb.type	Sybase_ASA	The monitoring database type.
cluster.version	1	The cluster version number.
cluster.sync.share-dpath	none	The shared data path for CSYNC/DSYNC zip files; required only when "cluster.sync.share-dpath.enabled" is set to true.
cluster.sync.share-dpath.enabled	False	Indicates whether or not to enable the cluster's optional "Shared Data Path" feature.
license.product.edi-tion	none	The Unwired Platform license edition.
license.type	none	The Unwired Platform license type.
client.licenses	none	The number of Unwired Platform client device licenses.

Property	Default	Description
client.url_suffix	none	the URL suffix used by a mobile client to connect to a Relay Server. For example, to connect a client to an Unwired Server cluster called RepFarm1, you might use `/sup_relay_server/client/rs_client.dll/RepFarm1` as the URL suffix.
msg.http.server.ports	5001, 80	The Messaging service HTTP ports.
msg.client.url_suffix	none	the URL suffix used by a mobile client to connect to a Relay Server. For example, to connect a client to an Unwired Server cluster called MsgFarm1, you might use `/sup_relay_server/client/rs_client.dll/MsgFarm1` as the URL suffix.
msg.admin.webservices.port	5100	The Messaging service administration port.
msg.rsoeOptions	none	The Messaging service RSOE options. Used in the Messaging service RSOE command line.
msgserver.location	none	The file location of the Messaging service.
webserver.rsoeOptions	none	(Applies only until previous versions are upgraded.) The RSOE options for embedded Web server requests.
webserver.client.url_suffix	none	the URL suffix used by a mobile client to connect to a Relay Server.The client URL suffix for Web server requests when the Relay Server is configured for embedded Web server requests..
rsoeOptions	none	(Applies only until previous versions are upgraded.) The RSOE options for the Messaging service. Used in the messaging protocol RSOE command line.
sqlany.mode	Primary	The cache database mode: only primary is supported.
mac.address	none	The MAC address of the network interface adapter used by the RSOE on the Unwired Server host.

Property	Default	Description
domainlogdb.server-host	none	The domain log database server host name.
domainlogdb.databasename	domainlogdb	The domain log database name.
domainlogdb.password	sql	The domain log database password.
domainlogdb.username	dba	The domain log database user name.
domainlogdb.server-port	5200	The domain log database server port.
domainlogdb.type	Sybase_ASA	The domain log database type.
ocsp.enable	none	Indicates whether OCSP checking is enabled when doing certificate revocation checking.
ocsp.responderCertSubjectName	none	The subject name of the OCSP responder's certificate, for example: `ocsp.responderCertSubjectName="CN=OCSP Responder, O=XYZ Corp"`
ocsp.responderCertIssuerName	none	The OCSP responder's certificate, for example: `ocsp.responderCertIssuerName="CN=Enterprise CA, O=XYZ Corp"`
ocsp.responderURL	none	The URL that identifies the location of the OCSP responder, for example: `ocsp.responderURL=http://ocsp.example.net:80`
ocsp.responderCertSerialNumber	none	The serial number of the OCSP responder's certificate, for example: `ocsp.responderCertSerialNumber=2A:FF:00`

Admin Security (default.xml) Configuration File Reference

Defines authentication and attribution properties for the 'admin' security configuration. The file is located in *<UnwiredPlatform_InstallDir>*`\Servers\UnwiredServer\Repository\CSI\conf`.

Note: Sybase recommends that you use Sybase Control Center to configure security settings, so that configuration changes are validated before being saved.

Define the login modules used for authentication requests. List the modules in the order that they are invoked with this syntax:

```
<config:authenticationProvider name="<authProviderName>"
controlFlag="<myValue>" />
```

See the *controlFlag* reference topic for possible configuration values. The default is `required`.

Configure global options if the same configuration is shared by the authentication and attribution providers by using:

```
<config:options name="<propertyName>" value="<myValue>" />
```

For an LDAP security provider, properties can include:

Property	Default	Description
ServerType	None	The LDAP server type. For example, `msad2k`, `sunone5`, `nsds4`, or `openldap`.
ProviderURL	`ldap://<host-name>:389`	The URL to connect to the LDAP server. For example, `ldap://localhost:389`. For SSL, use `ldap://localhost:636`.
SecurityProtocol	None	The protocol used to connect to the LDAP server. Sybase recommends that you set this to `SSL`. An SSL connection is required for Active-Directory when you set the password attribute.
InitialContextFactory	None	The factory class used to obtain initial directory context.
Referral	ignore	The behavior when a referral is encountered. The valid values are those dictated by LdapContext, for example, `ignore`, `follow`, or `throw`.
DefaultSearchBase	None	The default search base to use when performing general operations.

Property	Default	Description
AuthenticationSearchBase	None	The search base to use when performing authentication operations. If you do not set this value, the default search base is used.
SelfRegistrationSearchBase	None	The search base to use when creating a new user as part of self-registration. If you do not set this value, the authentication search base is used for self-update operations and the default search base is used for all other operations.
AuthenticationScope	ONELEVEL	Options include `ONELEVEL` or `SUBTREE`.
AuthenticationFilter	For most LDAP servers: `(&(uid={uid})(objectclass=person))` or For Active Directory e-mail lookups: `(&(userPrincipalName={uid})(objectclass=user)) [ActiveDirectory]` For Active Directory Windows user name lookups: `(&(sAMAccountName={uid})(objectclass=user))`	The user name and password authentication search filter. This must be a legal LDAP search filter, as defined in RFC 2254. The filter may contain the special string `"{uid}"` which is replaced with the user name of the user attempting to authenticate.

Property	Default	Description
CertificateAuthentication-Filter	For Active Directory server: `(&({certattr}={0})(object-class=user))` For most LDAP server types: `"(&({certattr}={0})(objectclass=person))"`	The certificate authentication search filter. The filter may contain the special string `"{certattr}"` which is replaced with the certificate attribute or the mapped LDAP attribute (if mapping between certificate attributes and LDAP attributes is defined). The value `"{0}"` is set to the encoded certificate or an attribute value from the certificate.
AuthenticationMethod	simple	The authentication method to use for all binding. Supported values are `DIGEST-MD5` or `simple`.
Attributes	None	Defines an attribute mapping from a CSI attribute to an LDAP attribute, including: • `CSI.Email` • `CSI.Username` • `CSI.Password`
BindDN	None	The DN to bind against when building the initial LDAP connection. The user being authenticated with the login module must have read permission on all user records.
BindPassword	None	The password to bind with for the initial LDAP connection. For example, `<config:options name="BindPassword" encrypted="true" value="1-AAAAEgQ-QyyYzC2+njB4K4QGPcMB1pM6XErTqZ1InyYrW/s56J69VfW5iBdFZeh-DrY66+6g9u1+a5VAqBiv/v5q08B3f59YMB1EQx9k93VgVTSC0w8q0="/>`.

Property	Default	Description
RoleSearchBase	None	The search base used to retrieve lists of roles. If this you do not set this value, the default search base is used.
RoleFilter	For SunONE/iPlanet: `(&(objectclass=ldapsubentry) (objectclass=nsroledefinition))` For Netscape Directory Server: `(objectclass=groupofnames) (objectclass=groupofuniquenames))` For ActiveDirectory: `(objectclass=groupofnames) (objectclass=group))`	When combined with the role search base and role scope, returns a complete list of roles within the LDAP server.
RoleScope	ONELEVEL	The role search scope. Options include `ONELEVEL` or `SUBTREE`.
RoleNameAttribute	cn	The attribute for retrieved roles that is the common name of the role.
UserFreeformRoleMembershipAttributes	None	The "freeform" role membership attribute list. Users who have attributes in this comma-delimited list are automatically granted access to roles that have names that are the same as the attribute value.

Property	Default	Description
UserRoleMembershipAttributes	The default value of this property depends on the chosen server type: • For SunONE 5.x, it is `"nsRoleDN"` • For ActiveDirectory, it is `"memberOf"`.	Defines the attributes that contain the DNs of all of the roles the user is a member of. These comma-delimited values are then cross-referenced with the roles retrieved from the role search base and search filter to create a list of user roles.
RoleMemberAttributes	There is a default value only for Netscape 4.x server: `"member,uniquemember"`	A comma-delimited list of potential attributes for roles. This list defines the DNs of people who are granted the specified roles. These values are cross-referenced with the active user to determine the user's roles.

Configure additional security providers using:

```
<config:provider name="<secProviderName>" type="<secType>" />
```

Security types include: `authorizer`, `attributer`, or `roleMapper`.

controlFlag Attribute Values

(Not applicable to Online Data Proxy) The Sybase implementation uses the same controlFlag values and definitions as those defined in the JAAS specification.

If you stack multiple providers, you must set the controlFlag attribute for each enabled provider.

Control Flag Value	Description
(Default) required	The LoginModule is required. Authentication proceeds down the LoginModule list.
requisite	The LoginModule is required. Subsequent behavior depends on the authentication result: • If authentication succeeds, authentication continues down the LoginModule list. • If authentication fails, control returns immediately to the application (authentication does not proceed down the LoginModule list).

Control Flag Value	Description
sufficient	The LoginModule is not required. Subsequent behavior depends on the authentication result: • If authentication succeeds, control returns immediately to the application (authentication does not proceed down the LoginModule list). • If authentication fails, authentication continues down the LoginModule list.
optional	The LoginModule is not required. Irrespective of success or failure, authentication proceeds down the LoginModule list.

Example

Providers are listed in this order and with these controlFlag:

1. CertificateAuthenticationLoginModule (sufficient)
2. LDAP (optional)
3. NativeOS (sufficient)

A client doing certificate authentication (for example, X.509 SSO to SAP) can authenticate immediately. Subsequent modules are not called, because they are not required. If there are regular username/password credentials, they go to LDAP, which may authenticate them, and set them up with roles from the LDAP groups they belong to. Then NativeOS is invoked, and if that also succeeds, Unwired Platform picks up roles based on the Windows groups they are in.

Unwired Server Logging (logging-configuration.xml) Configuration File

Defines the server logging configuration. This file is located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\Repository`

Note: Manual changes to the server log settings in one node of a cluster affect only the node on which the change is made. To modify cluster-wide server logging, you must individually edit the log file for each node.

Edit the `logging-configuration.xml` file to change server logging levels by component and define server file settings. The file syntax is:

```
<Entity EntityTypeId="<Logging Configuration>"
 <Entity EntityTypeId="LocalFileAppender">
  <EntityEntityTypeId="<UnwiredPlatformComponent>">
   <Prop name="LogLevel" value="<myLogLevel>" />
  </Entity>
  <Prop name="filename" value="<myFileName>"
  <Prop name="sizeRollover" value="<mySizeRollover>"
  <Prop name="maximumRolloverFiles" value="<myMaxRolloverFiles>"
  <Prop name="dateRollover" value="<myDateRollover>"
```

```
 </Entity>
</Entity>
```

The configurable properties are:

Property	Default	Description
LogLevel	The default log level varies depending upon the component: • Trace – TRACE • MMS – INFO • MSG – INFO • Security – INFO • MobiLink – INFO • DataServices - INFO • Other – WARN **Note:** If DOE-C is installed, it also appears as an Unwired Server component.	The log level for each configurable Unwired Platform component. Available log levels include: `ALL`, `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`, `OFF`, or `CONSOLE`. Trace is an interval component; do not modify it.
filename	logs/${sup.host}-server.log	The file name to which log data is written. Files are resolved relative to the server home directory.
sizeRollover	10MB	The number of bytes of granularity between rollover events. Supports optional KB, MB, and GB suffixes.
maximumRolloverFiles	1	The maximum number of rollover files that is maintained. Specify -1 for no limit.
dateRollover	NONE	The level of time granularity between rollover events: `NONE`, `HOURLY`, `DAILY`, `WEEKLY`, `MONTHLY`, `YEARLY`.

Runtime Message Tracing (TraceConfig.xml) Configuration File

Enables message tracing for various system components, and sets the tracing level. This file is located in `<UnwiredPlatform_InstallDir>\Servers\MessagingServer\Data`

The syntax is:

```
<TraceConfig>
            <Dir>\...UnwiredServer\logs</Dir>
            <Module Name="<moduleName>" Level="<traceLevel>"
Desc="<moduleDescription>" />
```

```
            ...
</TraceConfig>
```

The `<Dir>` property indicates the directory where tracing messages are stored when component tracing is enabled. The default location is `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\logs\<moduleName>`. Messages for each module are stored as .txt files in the corresponding folder.

The configurable properties are:

- Module name – the name of the system component.
- Level – the tracing level:

Level	Messages logged
All	Complete system information
Trace	Finer-grained informational events than debug
Debug	Very fine-grained system information, warnings, and all errors
Info	General system information, warnings, and all errors
Warn	Warnings and all errors
Error	Errors only
Off	None

- Desc – a unique identifying description for the module.

Sybase Control Center Configuration Files

Use Sybase Control Center configuration files to configure Sybase Control Center services, plug-ins, logging, and security.

Sybase recommends that you edit these `.properties` and `.xml` files only if you cannot use Sybase Control Center to configure the properties.

See also

- *Unwired Server Configuration Files* on page 275
- *Relay Server Configuration Files* on page 291

Sybase Control Center Services (service-config.xml) Configuration Files

Configure properties for various Sybase Control Center services. These files are located in `<UnwiredPlatform_InstallDir>\SCC-XX\services\<serviceName>`.

Key `service-config.xml` files include:

File	Description	Defaults
<UnwiredPlatform_InstallDir>\SCC-*XX*\services\RMI\service-config.xml	Sets the RMI agent port.	RMI port: 9999
<UnwiredPlatform_InstallDir>\SCC-*XX*\services\Messaging\service-config.xml	Sets the JMS messaging service port.	JMS messaging port: 2100
<UnwiredPlatform_InstallDir>\SCC-*XX*\services\SccSADataserver\service-config.xml	Sets the Sybase Control Center repository database port, and other properties.	Repository database port: 3683
<UnwiredPlatform_InstallDir>\SCC-*XX*\services\EmbeddedWebContainer\service-config.xml	Sets the Web container ports.	HTTP port: 8282 HTTPS port: 8283

Agent Plug-in Properties (agent-plugin.xml) Configuration File

Defines server properties for each Unwired Server to administer from Sybase Control Center. The file is located in *<UnwiredPlatform_InstallDir>*\SCC-*XX*\plugins\com.sybase.supadminplugin.

Properties include:

Property	Default	Description
auto.discovery.enable.on.startup	Personal Developer Edition: false Other editions: true	Enables or disables the automatic discovery of Unwired Servers when Sybase Control Center starts.

Property	Default	Description
auto.discovery.log.repeat	3	Repeats the logging of errors that are generated when a discovered Unwired Server is pinged. After the specified count is reached, Sybase Control Center *X.X* stops printing error message to the log, but continues to ping the discovered server.
auto.discovery.schedule.enable.on.startup	true	Enables or disables scheduled Unwired Server discoveries. Scheduled discovery allows the agent to periodically check for newly installed Unwired Servers.
auto.discovery.schedule.interval	300000	Sets the scheduled discovery interval (in milliseconds).
sup.server.path	`<UnwiredPlatform_InstallDir>\Servers\UnwiredServer`	Sets the installation for Unwired Server. The path must be fully-qualified.
auto.register.localSUP.enable	true	If true, automatically registers the server as a managed resouce in Sybase Control Center. After launching and authenticating, registered resources automatically appear in the Unwired Platform console. If false, you need to manually register the server as managed resource in Sybase Control Center.
server.edition	none	The local Unwired Server edition: • ED – enterprise developer edition • PD – personal developer edition

Sybase Control Center Logging (log4j.properties) Configuration File

Enables and configures Sybase Control Center logging. The log4j configuration file is located in `<UnwiredPlatform_InstallDir>\SCC-XX\conf`.

log4j properties include:

Property	Default	Description
log4j.appender.agent.File	`<UnwiredPlatform_InstallDir>\SCC-XX\log\agent.log`	The name and location of the Sybase Control Center log file.
log4j.appender.agent.MaxFileSize	5MB	The maximum size that a file can reach before a new one is created.
log4j.appender.agent.MaxBackupIndex	20	The number of log files that are backed up before the oldest file is deleted.

Relay Server Configuration Files

Use Relay Server configuration files to set up Relay Servers. Use RSOE configuration files to set up Outbound Enablers.

See also

- *Unwired Server Configuration Files* on page 275
- *Sybase Control Center Configuration Files* on page 288

Relay Server Configuration (rs.config)

The `rs.config` file defines the configuration of individual Relay Servers, and Relay Server clusters (farms).

The rs.config file is divided into four sections:

- **[relay_server] –** defines configuration for a single Relay Server
- **[backend_farm] –** defines a homogeneous group (e.g., a cluster) of back-end servers, which are targeted by client requests
- **[backend_server] –** defines the connection to each back-end server, supported by one or more RSOEs
- **[options] –** defines general configuration properties that apply to all Relay Servers in a cluster (farm)

[relay_server] Section

- **enable –** Specifies whether the Relay Server is included in a Relay Server cluster.

Possible values are:

- yes
- no

Default is `yes`.

- **host –** Host name or IP address to be used by the Outbound Enabler to make a direct connection to the Relay Server.
- **http_port –** HTTP port to be used by the Outbound Enabler to make a direct connection to the Relay Server.

  Possible values are:

  - 0 or off — Disable HTTP access from Outbound Enabler
  - 1 to 65535 — Enable HTTP access on the specified port

  Default is `80`.
- **https_port –** HTTPS port to be used by the Outbound Enabler to make a direct connection to the Relay Server.

  Possible values are:

  - 0 or off — Disable HTTP access from Outbound Enabler
  - 1 to 65535 — Enable HTTP access on the specified port

  Default is `443`.
- **description –** User-definable description of the Relay Server, up to 2048 characters.

[backend_farm] Section

- **active_cookie –** Specifies whether a cookie is set to maintain client-server session affinity.

  Possible values are:

  - yes — Relay Server injects a standard HTTP set-cookie command, with a proprietary cookie name in the response.
  - no — Use this option when the back-end farm serves a sessionless browser application.
- **active_header –** Specifies whether a header is set to maintain client-server session affinity.

  Possible values are:

  - yes — Relay Server injects a proprietary header in the response, in case intermediaries tamper with the active cookie.
  - no — Use this option to reduce traffic volume, if the back-end farm serves only browser applications, or if the active cookie works for all clients of the back-end farm.
- **backend_security –** Specifies the level of security required of an Outbound Enabler in the back-end farm.

  Possible values are:

  - on — All connections from the back-end farm must use HTTPS.
  - off — All connections from the back-end farm must use HTTP.

  If no value is specified, the Outbound Enabler can use either HTTP or HTTPS.
- **client_security –** Specifies the level of security the back-end farm requires of its clients.

  Possible values are:

  - on — All clients must use HTTPS.
  - off — All clients must use HTTP.

  If no value is specified, the clients can use either HTTP or HTTPS.
- **description –** User-definable description of the back-end farm, up to 2048 characters.
- **enable –** Specifies whether to allow connections from the back-end farm.

  Possible values are:

  - yes
  - no

  Default is `yes`.
- **id –** User-definable string that identifies the back-end farm, up to 2048 characters.
- **verbosity –** Relay Server logging verbosity.

  Possible values are:

  - 0 — Log errors only.
  - 1 — Session-level logging.
  - 2 — Request-level logging.
  - 3 - 5 — Detailed logging, used primarily for technical support.

  Default is `0`.

[backend_server] Section

- **description –** User-definable description of the back-end server, up to 2048 characters.
- **enable –** Specifies whether to allow connections from the back-end server.

  Possible values are:

  - yes
  - no

  Default is `yes`.
- **farm –** User-definable string that identifies a back-end farm.

  This property associates the back-end server with a back-end farm. This value must match the id value in a [backend_farm] section.
- **id –** User-definable string that identifies the back-end server, up to 2048 characters.

- **MAC –** MAC address of the network adapter used by Outbound Enabler to make a connection with the Relay Server.

  If this property is not specified, Relay Server does not check the MAC address on Outbound Enabler connections.
- **token –** A security token used by Relay Server to authenticate the back-end server connection, up to 2048 characters.
- **verbosity –** Relay Server logging verbosity.

  Possible values are:

  - 0 — Log errors only.
  - 1 — Session-level logging.
  - 2 — Request-level logging.
  - 3 - 5 — Detailed logging, used primarily for technical support.

  Default is `0`.

[options] Section

- **start –** Method used to start the State Manager.

  Possible values are:

  - auto — State Manager is started automatically by Relay Server Web extensions.
  - no — State Manager is started as a service.
  - full path — Full path to the State Manager executable (`rshost`).
  -

  Default is `auto`.
- **shared_mem –** Maximum amount of shared memory used for state tracking.

  Default is 10MB.
- –
- **verbosity –** Relay Server logging verbosity.

  Possible values are:

  - 0 — Log errors only.
  - 1 — Session-level logging.
  - 2 — Request-level logging.
  - 3 - 5 — Detailed logging, used primarily for technical support.

  Default is `0`.

Example: N+2 Architecture Configuration

There are two Relay Servers in a Relay Server farm: RS1.sybase.com and RS2.sybase.com. There are also two back-end server types: Afaria and Unwired Server. The Afaria system has

two servers: Afaria1.sybase.com and Afaria2.sybase.com. There is one Unwired Server: SUP.sybase.com.

Relay Servers use Microsoft IIS, and the Web server is configured with the following paths:

- `\ias_relay_server\client` – the install location of `rs_client.dll`.
- `\ias_relay_server\server` – the install location of `rs_server.dll`, `rshost.exe`, and `rs.config`.

`#----------------------------`
`# Relay server with auto start option`
`#----------------------------------------`
`[options]`
`start = no`
`verbosity = 1`
	Start State Manager as a Windows service.

`#--------------------`
`# Relay server peers`
`#--------------------`
`[relay_server]`
`enable = yes`
`host = RS1.sybase.com`
`http_port = 80`
`https_port = 443`
`description = Machine #1 in RS farm`
 	
`[relay_server]`
`enable = yes`
`host = RS2.sybase.com`
`http_port = 80`
`https_port = 443`
`description = Machine #2 in RS farm`
	Because there are two Relay Servers, both instances need to be defined in the peer list. This list is used by the RSOE to establish a connection with each Relay Server node in the farm.

`#---------------`
`# Backend farms`
`#---------------`
`[backend_farm]`
`enable          = yes`
`id              = AfariaFarm`
`client_security = on`
`backend_security= on`
`description     = This is the definition for the Afaria farm.`
 	
`[backend_farm]`
`enable          = yes`
`id              = SUPFarm`
`client_security = on`
`backend_security= on`
`description     = This is the definition for the SUP farm.`
	There are two types of back-end farms in this example. Unwired Server and Afaria. Each farm requires an entry in the [backend_farms] section, and each must have a unique Farm ID.

`#-----------------`
`# Backend servers`
`#-----------------`
`[backend_server]`
`enable   = yes`
`farm     = AfariaFarm`
`id       =  for Afaria1.syb-ase.com`
`mac      = 01-23-45-67-89-ab`
`token    = 7b2493b0-d0d4-464f-b0de-24643e1e0feb`
 	
`[backend_server]`
`enable   = yes`
`farm     = AfariaFarm`
`id       =  for Afaria1.syb-ase.com`
`mac      = 01-23-45-67-89-ac`
`token    = de1aac83-a653-4e0f-8a6c-0a161a6ee407`
 	
`[backend_server]`
`enable   = yes`
`farm     = SUPFarm`
`id       =  for SUP.sybase.com`
`mac      = 01-23-45-67-89-ad`
`token    = 621ece03-9246-4da7-99e3-c07c7599031c`
	There are three back-end server nodes in this scenario. Two are part of the AfariaFarm and the third is part of the SUPFarm. Afaria back-end servers use the Transmitter ID as the Server ID. Unwired Platform typically uses the machine name as the Server ID. The Server ID must be unique for each back-end server entry.

Note: When Relay Servers have been deployed and configured, clients connect to severs in the back-end farms differently:

- For the AfariaFarm example, clients would use:

```
url_suffix=/ias_relay_server/client/rs_client.dll/AfariaFarm
```

- For the SUPFarm example, clients would use:
  ```
  url_suffix=/ias_relay_server/client/rs_client.dll/SUPFarm
  ```

Outbound Enabler Configuration (rsoeconfig.xml)

The `rsoeconfig.xml` file defines Outbound Enabler (RSOE) and Relay Server configurations.

This information describes the XML schema of the RSOE configuration file.

Element: relayServers

XML root element.

Parents	n/a
Model	relayServer*, proxyServer*

Attribute	Value	Required
xmlns	Type: string http://www.sybase.com/sup/SUPRelayServerConfig	Yes

Element: relayServer

Identifies a Relay Server instance.

Parents	//relayServers
Model	description{0,1}, httpCredential*, unwiredServerFarm*

Attribute	Value	Required
ID	Type: integer Primary key of the Relay Server entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
host	Type: string Host name of the Relay Server, or host name of the load balancer (if any).	Yes
port	Type: non-negative integer, 0-65535 Relay Server HTTP port.	Yes

Attribute	Value	Required
securePort	Type: non-negative integer, 0-65535 Relay Server HTTPS port.	Yes
urlSuffix	Type: string URL suffix used by an RSOE to connect to the Relay Server.	Yes

Element: proxyServer

Identifies an Internet proxy server instance.

Parents	//relayServers
Model	proxyUser*

Attribute	Value	Required
guid	Type: ID Identifies the IP address and port of the proxy server, in this form: `IPaddress_port` For example: `10.56.225.169_808`	No
ID	Type: integer Primary key of the proxy server entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
host	Type: string Host name of the proxy server.	Yes
port	Type: non-negative integer, 0-65535 Proxy server HTTP port.	Yes

Element: description

User-definable description of a Relay Server, or an Unwired Server cluster.

Parents	//relayServers/relayServer //relayServers/relayServer/unwiredServerFarm
Model	n/a (text node only)

Attribute	Value	Required
n/a		

Element: httpCredential
Specifies credentials an RSOE must use to authenticate its connection to the Relay Server.

Parents	//relayServers/relayServer
Model	n/a (empty)

Attribute	Value	Required
ID	Type: integer Primary key of the credential entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
userName	Type: string User name RSOE must use to access the Relay Server.	Yes
password	Type: string Password RSOE must use to access the Relay Server.	Yes

Element: unwiredServerFarm
Identifies an Unwired Server cluster.

Parents	//relayServers/relayServer
Model	description{0,1}, serverNode*

Attribute	Value	Required
ID	Type: integer Primary key of the server farm entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
name	Type: string Name of the Unwired Server cluster, as known to the Relay Server.	Yes

Attribute	Value	Required
type	Type: string, enumerated Type of Unwired Server connection. Allowed values are: • messaging • replication	Yes

Element: proxyUser
Specifies credentials an RSOE must use to authenticate its connection to an Internet proxy server.

Parents	//relayServers/proxyServer
Model	n/a (empty)

Attribute	Value	Required
guid	Type: ID Identifies the IP address and port of the proxy server, with an appended user identifier, in this form: `IPaddress_port_UserId` For example: `10.56.225.169_808_User-001`	No
ID	Type: integer Primary key of the credential entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
userName	Type: string User name RSOE must use to access the proxy server.	Yes
password	Type: string Password RSOE must use to access the proxy server.	Yes

Element: serverNode
Identifies an Unwired Server instance in the Unwired Server cluster.

Parents	//relayServers/relayServer/unwiredServerFarm
Model	rsoe{0,1}

Attribute	Value	Required
ID	Type: integer Primary key of the server node entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
name	Type: string Name of the Unwired Server instance, as known to the Relay Server.	Yes
token	Type: string Token string RSOE must use to authenticate its connection to the Relay Server.	Yes

Element: rsoe

Identifies an Outbound Enabler (RSOE) associated with an Unwired Server instance.

Parents	//relayServers/relayServer/unwiredServerFarm/serverNode
Model	supServerPort, clusterMember, tlsOptions{0,1}

Attribute	Value	Required
ID	Type: integer Primary key of the RSOE entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
httpUser	Type: string If specified, this value should match the value of a //relay-Server/httpCredential/@userName instance.	No
proxyServerRef	Type: IDREF If specified, this value must match the value of a //proxyServ-er/@guid instance.	No
proxyUserRef	Type: IDREF If specified, this value must match the value of a //proxyServ-er/proxyUser/@guid instance.	No

Attribute	Value	Required
startOptions	Type: string Defines RSOE command-line startup options. If specified, this value must match the pattern: `\s*(((-v\s+[0-5])|(-d\s+\d+)|(-os\s+(1024\d|102[5-9]\d|10[3-9]\d{2}|1[1-9]\d{3}|[2-9]\d{4}|[1-9]\d{5,}))|(-ot))\s+)*((-v\s+[0-5])|(-d\s+\d+)|(-os\s+(1024\d|102[5-9]\d|10[3-9]\d{2}|1[1-9]\d{3}|[2-9]\d{4}|[1-9]\d{5,}))|(-ot))?\s*`	No
useHttps	Type: boolean Specifies whether RSOE uses HTTP or HTTPS in connection to Relay Server.	Yes

Element: supServerPort
Identifies the Unwired Server port managed by an RSOE.

Parents	//relayServers/relayServer/unwiredServerFarm/serverNode/rsoe
Model	n/a (empty)

Attribute	Value	Required
port	Type: non-negative integer, 0-65535	Yes

Element: clusterMember
Identifies the name of the Unwired Server instance associated with the RSOE.

Parents	//relayServers/relayServer/unwiredServerFarm/serverNode/rsoe
Model	n/a (empty)

Attribute	Value	Required
name	Type: string Name of the Unwired Server instance, as known to the Unwired Server cluster.	Yes

Element: tlsOptions
Specifies TLS configuration for RSOE connection to Relay Server.

Parents	//relayServers/relayServer/unwiredServerFarm/serverNode/rsoe
Model	n/a (empty)

Attribute	Value	Required
certificateCompany	Type: string Organization name field of certificate.	No
certificateFile	Type: anyURI Location of certificate file.	Yes
certificateName	Type: string Common name field of certificate.	No
certificateUnit	Type: string Organization unit field of certificate.	No
tlsType	Type: string, enumerated Allowed values are: • RSA	No

Content of rsoeConfig.template.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<relayServers xmlns="http://www.sybase.com/sup/
SUPRelayServerConfig">
    <!--
       If the "ID" attribute for a XML element is 0 means you want to
create
       a new element. If the "ID" attribute of a element is a positive
         integer means you want update a existing element.
    -->
    <relayServer securePort="443" port="80"
host="relayserver.sybase.com"
        urlSuffix="/ias_relay_server/server/rs_server.dll" ID="0">
       <description>Relay Server/RSOE Configuration example. A relay
            server could have many Backend Farms.
        </description>
        <unwiredServerFarm type="replication" name="farm1.myRBS"
ID="0">
           <description>Replication Backend Farm example. A Backend
Farm could have
                many Backend Servers.</description>
            <serverNode token="36413d78ef187a7b38548e00e586"
name="node1"
                ID="0">
              <!-- A Backend Server can have 0 or 1 RSOE mapping to
it. -->
```

```
             <rsoe useHttps="false" ID="0" startOptions="-ot -v 0 -d
10">
                  <supServerPort port="2480" />
                  <clusterMember name="ExampleServer1" />
               </rsoe>
            </serverNode>
            <serverNode token="123" name="node2" ID="0" />
            <serverNode token="my_secret_token" name="node3" ID="0">
               <rsoe startOptions="-v 0 -ot" useHttps="true" ID="0">
                  <supServerPort port="2480" />
                  <clusterMember name="ExampleServer3" />
                  <tlsOptions certificateFile="E:\tmp
\mms-1.6.0.1118\RsoeCerts\myserver.pem" />
               </rsoe>
            </serverNode>
        </unwiredServerFarm>
        <unwiredServerFarm type="messaging" name="farm2.myMBSFarm"
            ID="0">
            <description>test</description>
        </unwiredServerFarm>
    </relayServer>
    <relayServer securePort="443" port="80"
        host="relayserver.example.com" ID="0" urlSuffix="/srv/
iarelayserver">
        <description>test</description>
    </relayServer>
</relayServers>
```

Monitoring Database Schema

The monitoring database includes several tables from which information is retrieved by Sybase Control Center.

These system tables are accessible to any user. The contents of monitoring tables can be changed only by Unwired Platform components.

You can browse the contents of these tables by using any database browsing tool, for example, Sybase Central.

See also

- *Configuring Monitoring Performance Properties* on page 106

mms_rbs_request Table

Detailed history information for replication-based synchronization.

Column name	Column type	Description
id	integer	The identifier of the data row.

Column name	Column type	Description
summaryId	varchar(50)	The identifier for the row summary information.
deviceId	varchar(255)	The identifier of the physical device performing the synchronization request.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the MBO data synchronization or operation replay activity.
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
startTime	timestamp	The date and time the synchronization request was initiated.
endTime	timestamp	The date and time the synchronization request was completed.
syncTime	integer	The total time of the synchronization or operation replay activity, in milliseconds.
sendRows	integer	The number of rows downloaded during the mobile business object (MBO) synchronization. If 1 appears, the action was an operation replay.
isError	bit	Whether an error has occured during the synchronization or replay: 1 if errors were recorded, 0 if no errors were recorded.
sentBytes	integer	The number of bytes downloaded in the transaction.
receivedBytes	integer	The number of bytes uploaded in the transaction.
syncPhase	varchar(20)	The current synchronization activity: upload or download. During upload, a client initiates operation replays to execute MBO operations on the back-end system. During download, a client synchronizes with Unwired Server to receive the latest changes to an MBO from the back-end system.

Column name	Column type	Description
mboNames	varchar(500)	The MBO that downloaded information.
operationNames	varchar(500)	The operation replay.
operationReplays	integer	The number of operation replays performed. Zero (0) indicates that information was downloaded by the MBO during a synchronization action.
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
applicationId	varchar(100)	The identifier for the application information.

mms_rbs_request_summary Table

Summary history information for replication-based synchronization transactions.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device performing the synchronization request.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO) data synchronization or operation replay activity.
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
startTime	timestamp	The date and time the synchronization request was initiated.
endTime	timestamp	The date and time the synchronization request was completed.
request syncTime	integer	The total time of the synchronization or operation replay activity, in milliseconds.

Column name	Column type	Description
totalReceivedRows	integer	Always 1.
totalErrors	integer	The number of all exceptions during the synchronization request.
totalsentBytes	integer	The number of all bytes dowloaded by the MBO.
totalreceivedBytes	integer	The number of all bytes uploaded by the MBO.
totalOperationReplays	integer	The number of all operation replays for the MBO.
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
applicationId	varchar(100)	The identifier for the application information.
mbo	varchar(500)	The MBO that downloaded information.

mms_rbs_mbo_sync_info Table

A subset of the mms_rbs_request table data.

Column name	Column type	Description
id	integer	The identifier of the data row.
packageName	varchar(255)	The package name of the mobile business object (MBO) data synchronization.
domain	varchar(255)	The domain to which the package involved in synchronization belongs.
mboName	varchar(255)	The name of the MBO performing the transaction.
startTime	timestamp	The date and time the synchronization request was initiated.
endTime	timestamp	The date and time the synchronization request was completed.
syncTime	integer	The total time of the synchronization, in milliseconds.

Column name	Column type	Description
isError	bit	Whether errors have occured during the synchronization: 1 if errors were recorded, 0 if no errors were recorded.

mms_rbs_operation_replay Table

A subset of the mms_rbs_request table data.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device performing the operation replay.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO).
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
startTime	timestamp	The date and time the operation replay request was initiated.
endTime	timestamp	The date and time the operation replay was completed.
processTime	integer	The total time of the operation replay, in milliseconds.
mbo	varchar(255)	The MBO performing the transaction.
operation	varchar(255)	The operation performing the operation replay.
isError	bit	Whether errors occurred during the synchronization or replay: 1 if errors were recorded, 0 if no errors were recorded.

Column name	Column type	Description
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
applicationId	varchar(100)	The identifier for the application information.

mms_mbs_message Table

Detailed history information for message-based synchronization.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device performing the operation replay.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO).
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
receiveTime	timestamp	The received time of inbound message. Not applicable to outbound messages.
pushTime	timestamp	The pushed time of outbound message. Not applicable to inbound messages.
startTime	timestamp	The date and time the message was initiated.
endTime	timestamp	The date and time the message was completed.

Column name	Column type	Description
processTime	integer	The total time of the message, in milliseconds.
mboName	varchar(255)	The MBO performing the message.
operationName	varchar(255)	The operation performing the message.
messageType	varchar(50)	The type of message. One of: SUBSCRIBE, UNSUBSCRIBE, OPERATION_REPLAY, RECOVER, SUSPEND, RESUME, RESUMENOREPLAY, IMPORT_DATA, DATA_RESET, LOGIN, UNKNOWN_TYPE.
isError	bit	Value is 1 if errors were recorded during transaction. 0 if no errors recorded.
payloadSize	integer	The size of the message payload.
isPushMsg	bit	Value is 1 if the message is outbound, 1 if otherwise.
isRequestMsg	bit	Value is 1 if the message is inbound, 0 if otherwise.
isSubscription	bit	Value is 1 if the message is a subscription request, 0 if not.
isOperationReplay	bit	Value is 1 if the message is a message-based operation replay, 0 if not.
sentPayloadSize	integer	The payload size of the outbound message.
receivedPayloadSize	integer	The payload size of the inbound message.
isMonitored	bit	Value is 1 if MBO is monitored, 0 if not.

Column name	Column type	Description
isLogged	bit	Value is 1 if the domain is logging data, 0 if not.
applicationId	varchar(100)	The identifier for the application information.

mms_security_access Table

Information about security and user access.

Column	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device used during the authentication request.
userName	varchar(255)	The name of the user requesting authentication.
packageName	varchar(255)	The package name of the authentication request.
domain	varchar(255)	The domain to which the package belongs.
securityConfiguration	varchar(255)	The name of the security configuration performing the authentication.
access_time	timestamp	The time the request for access was made.
outcome	bit	The outcome of the authentication request: 1 means authentication passed; 0 means authentication failed.
reason	longvarchar	Reason for authentication failure.
applicationId	varchar(100)	The identifier for the application information.

mms_rbs_outbound_notification Table

Outbound notification information for replication-based synchronization packages.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device receiving the outbound notification.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the MBO.
domain	varchar(255)	The domain to which the package belongs.
publicationName	varchar(255)	The synchronization group of the outbound notification.
notificationTime	timestamp	The time the outbound notification was sent.
subscriptionId	integer	The identifier for the subscription.
subcriptionEnabled	bit	Whether the subscription is enabled. If 1, it is. If 0, it is not.
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
applicationId	varchar(100)	The identifier for the application information.

mms_data_change_notification Table

Information about data change notifications (DCNs) for messaging-based synchronization.

Column name	Column type	Description
packageName	varchar(255)	The package name of the mobile business object (MBO) affected by the DCN.
domain	varchar(255)	The domain to which the package involved in DCN belongs.
publicationName	varchar(255)	The synchronization group of the DCN.
notificationTime	timestamp	The time the DCN was sent.
processTime	integer	The total time to process the DCN, in milliseconds.
affectedRows	integer	The rows affected by the DCN.
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
mboName	varchar(500)	The MBO that downloaded information.

mms_concurrent_user_info Table

Information about concurrent users.

Column name	Column type	Description
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO) affected by the data change notification (DCN).
curTime	timestamp	The current time.

Column name	Column type	Description
domain	varchar(255)	The domain to which the package involved in DCN belongs.
type	bit	The type of request. If 1, it is a messaging request. If 0, it is a replication request.

mms_queue_info Table

Information about the messaging queue.

Colum	Column type	Description
queueName	varchar(255)	The name of the queue.
pendingItem	integer	The number of pendings messages in the server queue.
curTime	timestamp	The current time.

mms_sampling_time Table

Information about the sampling time.

Column	Column type	Description
sampling_time	timestamp	The sampling time
id	integer	The unique key of the table

cache_history Table

Saves the history events on the Unwired Server cache by a deployed package.

Column	Column type	Description
package_name	varchar(128)	The package name of the mobile business object.
activity_type	integer	The event by activity type: • 1=ON DEMAND FULL REFRESH • 2=ON DEMAND PARTITIONED REFRESH • 3=CACHE QUERY

Column	Column type	Description
cache_name	varchar(128)	The Unwired Server cache name.
mbo_name	varchar(128)	The mobile business object that triggered the activity.
start_time	datetime	The recorded start date and time of the activity.
duration	bigint	The recorded duration of the activity.
partition_key	varchar(128)	The partition key value.
host_name	varchar(64)	The host name.
process_id	varchar(64)	The process id.
unique_row_id	numeric(10,0)	Internal identifier only.

cache_history Stored Procedures

The cache_history table uses several stored procedures.

Procedure	Parameters	Result
`get_lastfullrefresh`	• packagename varchar(128) • cachename varchar(128)	The last full refresh value.
`get_lastinvalidatetime`	• packagename varchar(128) • cachename varchar(128)	The last invalid date value.
`get_lastupdatetime`	• packagename varchar(128) • cachename varchar(128)	The last update value.

cache_statistic Table

Saves the Unwired Server cache status details.

Column	Column type	Description
package_name	varchar(128)	The package name.
cache_name	varchar(128)	The cache name.

Column	Column type	Description
last_full_refresh	datetime	The last date and time a full cache refresh occurred.
last_update	datetime	The last date and time a cache update occurred.
last_invalidate	datetime	The last date and time an invalidate cache occurred.

cache_statistics Stored Procedures

Provide aggregations of cache activities over a date range for a mobile business object.

Procedure	Parameters	Result
`get_package_mbo_maxfullrefreshtime` `get_package_mbo_minfullrefreshtime`	• @packagename varchar(128) • mboname varchar (128) • startdate datetime • endate datetime	The minimum or maximum value of the duration for the mobile business object over the start date and end date range for a refresh activity.
`get_package_mbo_maxcachewaittime` `get_package_mbo_mincachewaittime`	• packagename varchar(128) • mboname varchar(128) • startdate datetime • endate datetime	The minimum or maximum value of the duration for the mobile business object over the start date and end date range for a cache activity.
`get_package_mbo_averageeachewaittime` `get_package_mbo_averagefullrefereshtime`	• packagename varchar(128) • cachename varchar(128) • startdate datetime • endate datetime	The average value of the duration for the mobile business object over the start date and end date range for a cache or a refresh activity.

Procedure	Parameters	Result
`get_package_mbo_access_count` `get_package_mbo_ondemandrefresh_count`	• packagename varchar(128) • cachename varchar(128) • startdate datetime • endate datetime	The count of access and on demand refresh activities for mobile businss object over the start date and end date range.

Application Connection Properties

Set application connection properties through Sybase Control Center to create application connections and application connection templates. These settings influence the application connection, including security, datasource connection, user registration, device and application, and so forth.

See also

- *Creating Application Connection Templates* on page 72
- *Application ID Overview* on page 70
- *Application ID Guidelines* on page 71
- *Setting General Application Properties* on page 68

Apple Push Notification Properties

Apple push notification properties allow iOS users to install client software on their devices. This process requires you to create a different e-mail activation message using the appropriate push notification properties.

- **APNS Device Token –** the Apple push notification service token. An application must register with Apple push notification service for the iOS to receive remote notifications sent by the application's provider. After the device is registered for push properly, this should contain a valid device token. See the iOS developer documentation.
- **Alert Message –** the message that appears on the client device when alerts are enabled. Default: `New items available`.
- **Delivery Threshold –** the frequency, in minutes, with which groupware notifications are sent to the device. Valid values: 0 – 65535. Default: 1.
- **Sounds –** indicates if a sound is a made when a notification is received. The sound files must reside in the main bundle of the client application. Because custom alert sounds are played by the iOS system-sound facility, they must be in one of the supported audio data formats. See the iOS developer documentation.

  Acceptable values: true and false.

Default: true

- **Badges –** the badge of the application icon.

  Acceptable values: true and false

  Default: true
- **Alerts –** the iOS standard alert. Acceptable values: true and false. Default: true.
- **Enabled –** indicates if push notification using APNs is enabled or not.

  Acceptable values: true and false.

  Default: true

Application Settings

Application settings display details that identify the Application Identifier, Domain, Security Configuration of an application connection template

- **Automatic Registration Enabled –** the value is set to **True** when the application connection registration is carried out automatically.
- **Application Identifier –** the application identifier registered on SCC.
- **Domain –** the domain selected for the connection template.
- **Security Configuration –** the security configuration defined for the connection template.

BlackBerry Push Notification Properties

BlackBerry push notification properties allow BlackBerry users to install messaging client software on their devices.

Property	Description
Enabled	Enables notifications to the device if the device is offline. This feature sends a push notification over an IP connection only long enough to complete the Send/Receive data exchange. BlackBerry Push notifications overcome issues with always-on connectivity and battery life consumption over wireless networks. Acceptable values: true (enabled) and false (disabled). If this setting is false, all other related settings are ignored. Default: true
Delivery threshold	The minimum amount of time the server waits to perform a push notification to the device since the previous push notification (in minutes). This controls the maximum number of push notifications sent in a given time period. For example, if three push notifications arrive 10 seconds apart, the server does not send three different push notifications to the device. Instead they are sent as a batch with no more than one push notification per X minutes (where X is the delivery threshold). Acceptable values: 0 – 65535. Default: 1

Property	Description
Push listener port	The push listener port reported by the device on which it listens for notifications. This port is automatically assigned by the client. For example, if there is another application already listening on this port, a free port is searched for. Default: 5011
Device PIN	Every Blackberry device has a unique permanent PIN. During initial connection and settings exchange, the device sends this information to the server. Unwired Server uses this PIN to address the device when sending notifications, by sending messages through the BES/MDS using an address such as: Device="Device PIN" + Port="Push Listener port". Default: 0
BES Notification Name	The BES server to which this device's notifications are sent. In cases where there are multiple BES servers in an organization, define all BES servers.

Connection Properties

Connection properties define the connection information used by Unwired Server to relate a user to a device.

- **Activation Code –** the original code sent to the user in the activation e-mail. Can contain only letters A – Z (uppercase or lowercase), numbers 0 – 9, or a combination of both. Acceptable range: 1 to 10 characters.
- **Farm ID –** a string associated with the Relay Server farm ID. Can contain only letters A – Z (uppercase or lowercase), numbers 0 – 9, or a combination of both. Default: 0.
- **Server Name –** the DNS name or IP address of the Unwired Server, such as "myserver.mycompany.com". If using Relay Server, the server name is the IP address or fully qualified name of the Relay Server host.
- **Server Port –** the port used for messaging connections between the device and Unwired Server. If using Relay Server, this is the Relay Server port. Default: 5001.
- **Synchronization Server Host –** the server host name used for synchronization.
- **Synchronization Server Port –** the port used for synchronization.
- **Synchronization Server Protocol –** the synchronization protocol - HTTP or HTTPS.
- **Synchronization Server Stream Parameters –** the server stream parameters.
- **Synchronization Server URL Suffix –** the server URL suffix.

Note: Some settings are visible but not in use by client (replication native application) at this time.

Custom Settings

Define one of four available custom strings that are retained during reregistration and cloning.

Change the property name and value according to the custom setting you require. The custom settings can be of variable length, with no practical limit imposed on the values. You can use

these properties to either manually control or automate how workflow-related messages are processed:

- Manual control – an administrator can store an employee title in one of the custom fields. This allows employees of a specific title to respond to a particular message.
- Automated – a developer stores the primary key of a back-end database using a custom setting. This key allows the database to process messages based on messaging device ID.

Device Advanced Properties

Advanced properties set specific behavior for messaging devices.

- **Relay Server URL Prefix –** the URL prefix to be used when the device client is connecting through Relay Server. The prefix you set depends on whether Relay Server is installed on IIS or Apache. Acceptable values:
  - For IIS – use`/ias_relay_server/client/rs_client.dll`.
  - For Apache – use `/cli/iasrelayserver`.
- **Allow Roaming –** the device is allowed to connect to server while roaming. Acceptable values: true and false. Default: true.
- **Debug Trace Size –** the size of the trace log on the device (in KB). Acceptable values: 50 to 10,000. Default: 50.
- **Debug Trace Level –** the amount of detail to record to the device log. Acceptable values: 1 to 5, where 5 has the most level of detail and 1 the least. Default: 1.
- **Device Log Items –** the number of items persisted in the device status log. Acceptable values: 5 to 100. Default: 50.
- **Keep Alive (sec) –** the Keep Alive frequency used to maintain the wireless connection, in seconds. Acceptable values: 30 to 1800. Default: 240.

Device Info Properties

Information properties display details that identify the mobile device, including International Mobile Subscriber identity (IMSI), phone number, device subtype, and device model.

- **IMSI –** the International Mobile Subscriber identity, which is a unique number associated with all Global System for Mobile communication (GSM) and Universal Mobile Telecommunications System (UMTS) network mobile phone users. To locate the IMSI, check the value on the SIM inside the phone.
- **Phone Number –** the phone number associated with the registered mobile device.
- **Device Subtype –** the device subtype of the messaging device. For example, if the device model is a BlackBerry, the subtype is the form factor (for example, BlackBerry Bold).
- **Model –** the manufacturer of the registered mobile device.

Proxy Properties

(Applies only to Online Data Proxy) Proxy properties display details that identify the application and push endpoints.

- **Application Endpoint –** the back-end URL where the application service document is available
- **Push Endpoint –** the server URL where all the notifications are forwarded to.

Security Settings

Security settings display the device security configuration.

- E2E Encryption Enabled – indicate whether end-to-end encryption is enabled or not: true indicates encryption is enabled; false indicates encryption is disabled.
- E2E Encryption Type – use RSA as the asymmetric cipher used for key exchange for end-to-end encryption.
- TLS Type – use RSA as the TLS type for device to Unwired Server communication.

Note: These settings are visible, but not in use by client (replication native application) at this time.

User Registration

User registration specifies details of the activation code that is sent to a user to manually activate an application on the device.

- Activation Code Expiration (Hours) – indicates how long an activation code is valid. The default is 72 hours.
- Activation Code Length – indicates the length of the activation code, as in number of alphanumeric characters. The default is 3.

Domain Log Categories

Domain log data provides detailed statistcs for all aspects of device, user, application, domain, and data synchronization related activities.

Synchronization Log

Synchronization logs include data related to different aspects of data synchronization, including data, subscriptions, operations, result checker, cache refresh and the data service and Unwired Server interface. Using data in these logs and the correlation tool, you can follow the data path between the enterprise information system (EIS), Unwired Server, cache database, and user application connection.

To find out about	See
Data synchronization transactions	Data Sync statistics
Data services requests made to the Enterprise information system (EIS)	DS Interface statistics
Cache database (CDB) activities	Cache refresh statistics
EIS error codes or failures resulting from Mobile Business Object operations against the EIS datasource	Result Checker statistics (coding required)
Moving MBO operations from a mobile device to the CDB	Operation replay statistics
Moving data between a mobile device and the CDB	Subscription statistics

Data Sync

Data Sync – basic statistics for individual data synchronizations:

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Sync Group – the synchronization group associated with the request.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Payload

Operation Replay

Operation Replay – statistics for moving MBO operations (typically create, update, and delete) from the device cache to the cache database cache on Unwired Server:

- Time – the time and date stamp for the log entry.

- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Payload

Subscription

Subscription – statistics for transferring data between mobile devices and the cache database on Unwired Server:

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- Subscription Type – the type of subscription used, including SUBSCRIBE, UNSUBSCRIBE, RECOVER, SUSPEND, and RESUME.
- Subscription ID – the identifier associated with the subscription.
- Sync Group – the synchronization group associated with the request.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Payload

Result Checker

Result Checker – EIS error codes or failures resulting from Mobile Business Object operations against the EIS datasource (requires coding):

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- Class – the class used for the result checker.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- None

Cache Refresh

Cache Refresh – statistics for cache database activities:

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Cache Group – the cache group name.
- CacheRow Count – the number of cached rows.
- EIS Row Count – the number of rows retrieved from the enterprise information system (EIS).
- Insert Count – the number of rows inserted in the cache.
- Update Count – the number of rows updated in the cache.
- Delete Count – the number of rows deleted from the cache.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Refresh Type
- Virtual Table Name

- Partition Key
- Pre Update Cache Image
- Post Update Cache Image

DS Interface

DS Interface – statistics for data services requests made to the Enterprise information system (EIS):

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Operation Type
- Virtual Table Name
- Input Attributes (payload)
- Input Parameters (payload)

Device Notification Log

Device notification logs include logging data for server-initiated synchronization notifications between Unwired Server and devices.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Sync Group – the synchronization group associated with the request.
- Thread ID – the identifier for the thread used to process the request.

- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Payload

Data Change Notification Log

Data Change Notification (DCN) logs include logging data for data change notifications between an enterprise information system (EIS) and an MBO package, for general and workflow DCN.

General Data Change Notification

For general DCN:

- Time – the time and date stamp for the log entry.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Payload

Workflow Data Change Notification

For workflow DCN:

- Time – the time and date stamp for the log entry.
- Workflow ID – the unique identifier associated with a workflow.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Package – the name of the package to which the subscription belongs.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Operation – the MBO operation.
- Subject – the workflow DCN request subject line.
- From – the "From" value for the workflow DCN request.
- To – the "To" value for the workflow DCN request.

- Body – the message body for the workflow DCN request.
- Error – the error message if any.

Note: Additional detail columns:

- Payload

Security Log

Security logs provide security details for individual applications, application connections, and users. Logs capture authentication failures and errors, and provide supporting information that identifies request-response messaging, package and MBO details, security configuration, and the thread and node that attempted to process an authentication request.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Correlation ID – the unique ID associated with every request-response message pair.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Security Configuration – the associated security configuration.
- Method – the MBO operation used.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Outcome – the authentication outcome for the security check.
- Reason – the reason for authentication failure.
- Error – the error message if any.

Error Log

Errors log data includes domain-level errors.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Correlation ID – the unique ID associated with every request-response message pair.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.

- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Connection Log

Connections log data includes domain connections for specific connection types to backend data sources, including DOE, JDBC, REST, SAP, and SOAP, if enabled. Additional columns may be available in the detail pane, and payload data may be available if enabled.

DOE Connection

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Event Type – the DOE-C event type, such as Acknowledged, Ignored, Exclude, Resend (from client), and Status.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Connection – the managed connection used.
- Client ID – the identifier for the DOE-C client.
- Physical ID – the DOE-C generated physical identifier registered with DOE at subscription.
- Subscription ID – the DOE-C generated subscription identifier registered with DOE at subscription.
- Logical Device ID – the DOE-C logical device identifier, generated by DOE and provided to DOE-C upon successful subscription.
- Message Direction – the DOE-C message direction, either client to Unwired Server, or Unwired Server to client.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Payload and detail columns:

- Endpoint Name (payload)
- JSON Message Content (payload)
- XML Message Content (payload)
- DOE server message ID
- DOE client message ID

- DOE-C server message ID
- DOE-C client message ID
- DOE-C method name
- DOE-C action name
- Push to
- Address
- Log
- Extract Window
- PBI

JDBC Connection

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Connection – the managed connection used.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Payload and detail columns:

- Input Parameters (payload)
- Query (payload)
- Endpoint Name
- Database Product Name
- Database Product Version
- Driver Name
- Driver Version
- Database User Name

REST Connection

- Time – the time and date stamp for the log entry.

- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Connection – the managed connection used.
- URL – the URL associated with the managed connection.
- Action – the GET, POST, PUT, or DELETE action.
- Response Status – the response status code for the invocation.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Payload and detail columns:

- Response (payload)
- Endpoint Name
- HTTP Header Parameters

SAP Connection

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- BAPI – the SAP BAPI used as the data source.
- Connection – the managed connection used.
- Properties – the list of name:value pairs.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Payload and detail columns:

- Parameters (payload)
- Endpoint Name
- SAP Host
- SAP User

SOAP Connection

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Connection – the managed connection used.
- Service Address – the service address URL.
- Action – the SOAP action.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Payload and detail columns:

- Request (payload)
- Response (payload)
- Endpoint Name
- Connection Timeout
- Authentication Type

Proxy Request-Response Log

Proxy Request-Response log data includes data for all requests and responses made from the Proxy server.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Source - the source of the log if its from the server or client.

- Correlation ID - the unique id associated with every request-response message pair.
- Request Type - the request type of the message.
- Request URL - the Gateway URL.
- HTTP Endpoint - the Gateway URL.
- Log Level - not relevant.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Post Data
- Request Header Fields
- Response Body
- Response Header Fields

Proxy Push Log

Proxy push logs include log data for all push notifications from the Proxy server.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Source - the source of the log if its from the server or client.
- Correlation ID - the unique id associated with every request-response message pair.
- URN - not relevant.
- Log Level - not relevant.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Additional detail columns:

- Message Body

APPENDIX B Glossary: Sybase Unwired Platform

Defines terms for all Sybase Unwired Platform components.

administration perspective – Or administration console. The Unwired Platform administrative perspective is the Flash-based Web application for managing Unwired Server. *See* Sybase Control Center.

administrators – Unwired Platform users to which an administration role has been assigned. A user with the "SUP Administrator" role is called a "platform administrator" and a user with the "SUP Domain Administrator" role is called a "domain administrator". These administration roles must also be assigned SCC administration roles to avoid having to authenticate to Sybase Control Center in addition to Unwired Server:

- A domain administrator only requires the "sccUserRole" role.
- A platform administrator requires both the "sccAdminRole" and "sccUserRole" roles.

Adobe Flash Player – Adobe Flash Player is required to run Sybase Control Center. Because of this player, you are required to run Sybase Control Center in a 32-bit browser. Adobe does not support 64-bit browsers.

Advantage Database Server® – A relational database management system that provides the messaging database for Sybase Unwired Platform. *See* messaging database.

Afaria® – An enterprise-grade, highly scalable device management solution with advanced capabilities to ensure that mobile data and devices are up-to-date, reliable, and secure. Afaria is a separately licensed product that can extend the Unwired Platform in a mobile enterprise. Afaria includes a server (Afaria Server), a database (Afaria Database), an administration tool (Afaria Administrator), and other runtime components, depending on the license you purchase.

application – In Unwired Server (and visible in Sybase Control Center), and application is the runtime entity that can be directly correlated to a native or mobile workflow application. The application definition on the server establishes the relationship among packages used in the application, domain that the application is deployed to, user activation method for the application, and other application specific settings.

APNS – Apple Push Notification Service.

application connection – A unique connection to the application on a device.

application connection template – a template for application connections that includes application settings, security configuration, domain details, and so forth.

application node – In Sybase Control Center, this is a registered application with a unique ID. This is the main entity that defines the behavior of device and backend interactions.

application registration – The process of registering an application with Sybase Unwired Platform. Registration requires a unique identity that defines the properties for the device and backend interaction with Unwired Server.

artifacts – Artifacts can be client-side or automatically generated files; for example: `.xml`, `.cs`, `.java`, `.cab` files.

availability – Indicates that a resource is accessible and responsive.

BAPI – Business Application Programming Interface. A BAPI is a set of interfaces to object-oriented programming methods that enable a programmer to integrate third-party software into the proprietary R/3 product from SAP®. For specific business tasks such as uploading transactional data, BAPIs are implemented and stored in the R/3 system as remote function call (RFC) modules.

BLOB – Binary Large Object. A BLOB is a collection of binary data stored as a single entity in a database management system. A BLOB may be text, images, audio, or video.

cache – The virtual tables in the Unwired Server cache database that store synchronization data. *See* cache database.

cache group – Defined in Unwired WorkSpace, MBOs are grouped and the same cache refresh policy is applied to their virtual tables (cache) in the cache database

cache partitions – Partitioning the cache divides it into segments that can be refreshed individually, which gives better system performance than refreshing the entire cache. Define cache partitions in Unwired WorkSpace by defining a partition key, which is a load argument used by the operation to load data into the cache from the enterprise information system (EIS).

cache database – Cache database. The Unwired Server cache database stores runtime metadata (for Unwired Platform components) and cache data (for MBOs). *See also* data tier.

CLI – Command line interface. CLI is the standard term for a command line tool or utility.

client application – *See* mobile application.

client object API – The client object API is described in the *Developer Guide: BlackBerry Native Applications*, *Developer Guide: iOS Native Applications*, and *Developer Guide: Windows and Windows Mobile Native Applications*.

cluster – Also known as a server farm. Typically clusters are setup as either runtime server clusters or database clusters (also known as a data tier). Clustering is a method of setting up redundant Unwired Platform components on your network in order to design a highly scalable and available system architecture.

cluster database – A data tier component that holds information pertaining to all Unwired Platform server nodes. Other databases in the Unwired Platform data tier includes the cache, messaging, and monitoring databases.

connection – Includes the configuration details and credentials required to connect to a database, Web service, or other EIS.

connection pool – A connection pool is a cache of Enterprise Information System (EIS) connections maintained by Unwired Server, so that the connections can be reused when Unwired Server receives future requests for data.

For proxy connections, a connection pool is a collection of proxy connections pooled for their respective back-ends, such as SAP Gateway.

connection profile – In Unwired WorkSpace, a connection profile includes the configuration details and credentials required to connect to an EIS.

context variable – In Unwired WorkSpace, these variables are automatically created when a developer adds reference(s) to an MBO in a mobile application. One table context variable is created for each MBO attribute. These variables allow mobile application developers to specify form fields or operation parameters to use the dynamic value of a selected record of an MBO during runtime.

data change notification (DCN) – Data change notification (DCN) allows an Enterprise Information System (EIS) to synchronize its data with the cache database through a push event.

data refresh – A data refresh synchronizes data between the cache database and a back-end EIS so that data in the cache is updated. *See also* scheduled data refresh.

data source – In Unwired WorkSpace, a data source is the persistent-storage location for the data that a mobile business object can access.

data tier – The data tier includes Unwired Server data such as cache, cluster information, and monitoring. The data tier includes the cache database (CDB), cluster, monitoring, and messaging databases.

data vault – A secure store across the platform that is provided by an SUP client.

deploy – (Unwired Server) Uploading a deployment archive or deployment unit to an Unwired Server instance. Unwired Server can then make these units accessible to users via a client application that is installed on a mobile device.

There is a one-to-one mapping between an Unwired WorkSpace project and a server package. Therefore, all MBOs that you deploy from one project to the same server are deployed to the same server package.

deployment archive – In Unwired WorkSpace, a deployment archive is created when a developer creates a package profile and executes the **build** operation. Building creates an archive that contains both a deployment unit and a corresponding descriptor file. A

deployment archive can be delivered to an administrator for deployment to a production version of Unwired Server.

deployment descriptor – A deployment descriptor is an XML file that describes how a deployment unit should be deployed to Unwired Server. A deployment descriptor contains role-mapping and domain-connection information. You can deliver a deployment descriptor and a deployment unit—jointly called a deployment archive—to an administrator for deployment to a production version of Unwired Server.

deployment mode – You can set the mode in which a mobile application project or mobile deployment package is deployed to the target Unwired Server.

deployment profile – A deployment profile is a named instance of predefined server connections and role mappings that allows developers to automate deployment of multiple packages from Sybase Unwired WorkSpace to Unwired Server. Role mappings and connection mappings are transferred from the deployment profile to the deployment unit and the deployment descriptor.

deployment unit – The Unwired WorkSpace build process generates a deployment unit. It enables a mobile application to be effectively installed and used in either a preproduction or production environment. Once generated, a deployment unit allows anyone to deploy all required objects, logical roles, personalization keys, and server connection information together, without requiring access to the whole development project. You can deliver a deployment unit and a deployment descriptor—jointly called a deployment archive—to an administrator for deployment to a production version of Unwired Server.

development package – A collection of MBOs that you create in Unwired WorkSpace. You can deploy the contents of a development package on an instance of Unwired Server.

device application – *See also* mobile application. A device application is a software application that runs on a mobile device.

device notification – Replication synchronization clients receive device notifications when a data change is detected for any of the MBOs in the synchronization group to which they are subscribed. Both the change detection interval of the synchronization group and the notification threshold of the subscription determine how often replication clients receive device notifications. Administrators can use subscription templates to specify the notification threshold for a particular synchronization group.

device user – The user identity tied to a device.

DML – Data manipulation language. DML is a group of computer languages used to retrieve, insert, delete, and update data in a database.

DMZ – Demilitarized zone; also known as a perimeter network. The DMZ adds a layer of security to the local area network (LAN), where computers run behind a firewall. Hosts running in the DMZ cannot send requests directly to hosts running in the LAN.

domain administrator – A user to which the platform administrator assigns domain administration privileges for one or more domain partitions. The domain administrator has a restricted view in Sybase Control Center, and only features and domains they can manage are visible.

domains – Domains provide a logical partitioning of a hosting organization's environment, so that the organization achieves increased flexibility and granularity of control in multitenant environments. By default, the Unwired Platform installer creates a single domain named "default". However the platform administrator can also add more domains as required.

EIS – Enterprise Information System. EIS is a back-end system, such as a database.

Enterprise Explorer – In Unwired WorkSpace, Enterprise Explorer allows you to define data source and view their metadata (schema objects in case of database, BAPIs for SAP, and so on).

export – The Unwired Platform administrator can export the mobile objects, then import them to another server on the network. That server should meet the requirement needed by the exported MBO.

hostability – *See* multitenancy.

IDE – Integrated Development Environment.

JDE – BlackBerry Java Development Environment.

key performance indicator (KPI) – Used by Unwired Platform monitoring. KPIs are monitoring metrics that are made up for an object, using counters, activities, and time which jointly for the parameters that show the health of the system. KPIs can use current data or historical data.

keystore – The location in which encryption keys, digital certificates, and other credentials in either encrypted or unencrypted keystore file types are stored for Unwired Server runtime components. *See also* truststore.

LDAP – Lightweight Directory Access Protocol.

local business object – Defined in Unwired WorkSpace, local business objects are not bound to EIS data sources, so cannot be synchronized. Instead, they are objects that are used as local data store on device.

logical role – Logical roles are defined in mobile business objects, and mapped to physical roles when the deployment unit that contain the mobile business objects are deployed to Unwired Server.

matching rules – A rule that triggers a mobile workflow application. Matching rules are used by the mobile workflow email listener to identify e-mails that match the rules specified by the administrator. When emails match the rule, Unwired Server sends the e-mail as a mobile workflow to the device that matches the rule. A matching rule is configured by the administrator in Sybase Control Center.

MBO – Mobile business object. The fundamental unit of data exchange in Sybase Unwired Platform. An MBO roughly corresponds to a data set from a back-end data source. The data can come from a database query, a Web service operation, or SAP. An MBO contains both concrete implementation-level details and abstract interface-level details. At the implementation-level, an MBO contains read-only result fields that contain metadata about the data in the implementation, and parameters that are passed to the back-end data source. At the interface-level, an MBO contains attributes that map to result fields, which correspond to client properties. An MBO may have operations, which can also contain parameters that map to arguments, and which determines how the client passes information to the enterprise information system (EIS).

You can define relationships between MBOs, and link attributes and parameters in one MBO to attributes and parameters in another MBO.

MBO attribute – An MBO attribute is a field that can hold data. You can map an MBO attribute to a result field in a back-end data source; for example, a result field in a database table.

MBO binding – An MBO binding links MBO attributes and operations to a physical data source through a connection profile.

MBO operation – An MBO operation can be invoked from a client application to perform a task; for example, create, delete, or update data in the EIS.

MBO relationship – MBO relationships are analogous to links created by foreign keys in a relational database. For example, the account MBO has a field called *owner_ID* that maps to the *ID* field in the owner MBO.

Define MBO relationships to facilitate:

- Data synchronization
- EIS data-refresh policy

messaging based synchronization – A synchronization method where data is delivered asynchronously using a secure, reliable messaging protocol. This method provides fine-grained synchronization (synchronization is provided at the data level—each process communicates only with the process it depends on), and it is therefore assumed that the device is always connected and available. *See also* synchronization.

messaging database – The messaging database allows in-flight messages to be stored until they can be delivered. This database is used in a messaging based synchronization environment. The messaging database is part of the Unwired Platform data tier, along with the cache, cluster, and monitoring databases.

mobile application – A Sybase Unwired Platform mobile application is an end-to-end application, which includes the MBO definition (back-end data connection, attributes, operations, and relationships), the generated server-side code, and the client-side application code.

Mobile Application Diagram – The Mobile Application Diagram is the graphical interface to create and edit MBOs. By dragging and dropping a data source onto the Mobile Application Diagram, you can create a mobile business object and generate its attribute mappings automatically.

Mobile Application Project – A collection of MBOs and client-side, design-time artifacts that make up a mobile application.

mobile workflow packages – Mobile workflow packages use the messaging synchronization model. The mobile workflow packages are deployed to Unwired Server, and can be deployed to mobile devices, via the Unwired Platform administrative perspective in Sybase Control Center.

monitoring – Monitoring is an Unwired Platform feature available in Sybase Control Center that allows administrators to identify key areas of weakness or periods of high activity in the particular area they are monitoring. It can be used for system diagnostic or for troubleshooting. Monitored operations include replication synchronization, messaging synchronization, messaging queue, data change notification, device notification, package, user, and cache activity.

monitoring database – A database that exclusively stores data related to replication and messaging synchronization, queues status, users, data change notifications, and device notifications activities. By default, the monitoring database runs in the same data tier as the cache database, messaging database and cluster database.

monitoring profiles – Monitoring profiles specify a monitoring schedule for a particular group of packages. These profiles let administrators collect granular data on which to base domain maintenance and configuration decisions.

multitenancy – The ability to host multiple tenants in one Unwired Cluster. Also known as hostability. *See also* domains.

node – A host or server computer upon which one or more runtime components have been installed.

object query – Defined in Unwired WorkSpace for an MBO and used to filter data that is downloaded to the device.

onboarding – The enterprise-level activation of an authentic device, a user, and an application entity as a combination, in Unwired Server.

operation – *See* MBO operation.

package – A package is a named container for one or more MBOs. On Unwired Server a package contains MBOs that have been deployed to this instance of the server.

palette – In Unwired WorkSpace, the palette is the graphical interface view from which you can add MBOs, local business objects, structures, relationships, attributes, and operations to the Mobile Application Diagram.

parameter – A parameter is a value that is passed to an operation/method. The operation uses the value to determine the output. When you create an MBO, you can map MBO parameters to data-source arguments. For example, if a data source looks up population based on a state abbreviation, the MBO gets the state from the user, then passes it (as a parameter/argument) to the data source to retrieve the information. Parameters can be:

- Synchronization parameters – synchronize a device application based on the value of the parameter.
- Load arguments – perform a data refresh based on the value of the argument.
- Operation parameters – MBO operations contain parameters that map to data source arguments. Operation parameters determine how the client passes information to the enterprise information system (EIS).

personalization key – A personalization key allows a mobile device user to specify attribute values that are used as parameters for selecting data from a data source. Personalization keys are also used as operation parameters. Personalization keys are set at the package level. There are three type of personalization keys: Transient, client, server.

They are most useful when they are used in multiple places within a mobile application, or in multiple mobile applications on the same server. Personalization keys may include attributes such as name, address, zip code, currency, location, customer list, and so forth.

perspective – A named tab in Sybase Control Center that contains a collection of managed resources (such as servers) and a set of views associated with those resources. The views in a perspective are chosen by users of the perspective. You can create as many perspectives as you need and customize them to monitor and manage your resources.

Perspectives allow you to group resources ways that make sense in your environment—by location, department, or project, for example.

physical role – A security provider group or role that is used to control access to Unwired Server resources.

Problems view – In Eclipse, the Problems view displays errors or warnings for the Mobile Application Project.

provisioning – The process of setting up a mobile device with required runtimes and device applications. Depending on the synchronization model used and depending on whether or not the device is also an Afaria client, the files and data required to provision the device varies.

pull synchronization – Pull synchronization is initiated by a remote client to synchronize the local database with the cache database. On Windows Mobile, pull synchronization is supported only in replication applications.

push synchronization – Push is the server-initiated process of downloading data from Unwired Server to a remote client, at defined intervals, or based upon the occurrence of an event.

queue – In-flight messages for a messaging application are saved in a queue. A queue is a list of pending activities. The server then sends messages to specific destinations in the order that

they appear in the queue. The depth of the queue indicates how many messages are waiting to be delivered.

relationship – *See* MBO relationship.

relay server – *See also* Sybase Hosted Relay Service.

resource – A unique Sybase product component (such as a server) or a subcomponent.

REST web services – Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

RFC – Remote Function Call. You can use the RFC interface to write applications that communicate with SAP R/3 applications and databases. An RFC is a standalone function. Developers use SAP tools to write the Advanced Business Application Programming (ABAP) code that implements the logic of a function, and then mark it as "remotely callable," which turns an ABAP function into an RFC.

role – Roles control access to Sybase Unwired Platform resources. *See also* logical role and physical role.

role mapping – Maps a physical (server role) to a logical (Unwired Platform role). Role mappings can be defined by developers, when they deploy an MBO package to a development Unwired Server, or by platform or domain administrators when they assign a security configuration to a domain or deploy a package to a production Unwired Server (and thereby override the domain-wide settings in the security configuration).

RSOE – Relay Server Outbound Enabler. An RSOE is an application that manages communication between Unwired Server and a relay server.

runtime server – An instance of Unwired Server that is running. Typically, a reference to the runtime server implies a connection to it.

SAP – SAP is one of the EIS types that Unwired Platform supports.

SCC – Sybase Control Center. A Web-based interface that allows you to administer your installed Sybase products.

schedule – The definition of a task (such as the collection of a set of statistics) and the time interval at which the task must execute in Sybase Control Center.

scheduled data refresh – Data is updated in the cache database from a back-end EIS, based on a scheduled data refresh. Typically, data is retrieved from an EIS (for example, SAP) when a device user synchronizes. However, if an administrator wants the data to be preloaded for a mobile business object, a data refresh can be scheduled so that data is saved locally in a cache. By preloading data with a scheduled refresh, the data is available in the information server when a user synchronizes data from a device. Scheduled data refresh requires that an administrator define a cache group as "scheduled" (as opposed to "on-demand").

security configuration – Part of the application user and administration user security. A security configuration determines the scope of user identity, authentication and authorization

checks, and can be assigned to one or more domains by the platform administrator in Sybase Control Center. A security configuration contains:

- A set of configured security providers (for example LDAP) to which authentication, authorization, attribution is delegated.
- Role mappings (which can be specified at the domain or package level)

security provider – A security provider and it's repository holds information about the users, security roles, security policies, and credentials used by some to provide security services to Unwired Platform. A security provider is part of a security configuration.

security profile – Part of the Unwired Server runtime component security. A security profile includes encryption metadata to capture certificate alias and the type of authentication used by server components. By using a security profile, the administrator creates a secured port over which components communicate.

server connection – The connection between Unwired WorkSpace and a back-end EIS is called a server connection.

server farm – *See also* cluster. Is the relay server designation for a cluster.

server-initiated synchronization – *See* push synchronization.

SOAP – Simple Object Access Protocol. SOAP is an XML-based protocol that enables applications to exchange information over HTTP. SOAP is used when Unwired Server communicates with a Web service.

solution – In Visual Studio, a solution is the high-level local workspace that contains the projects users create.

Solution Explorer – In Visual Studio, the Solution Explorer pane displays the active projects in a tree view.

SSO – Single sign-on. SSO is a credential-based authentication mechanism.

statistics – In Unwired Platform, the information collected by the monitoring database to determine if your system is running as efficiently as possible. Statistics can be current or historical. Current or historical data can be used to determine system availability or performance. Performance statistics are known as key performance indicators (KPI).

Start Page – In Visual Studio, the Start Page is the first page that displays when you launch the application.

structured data – Structured data can be displayed in a table with columns and labels.

structure object – Defined in Unwired WorkSpace, structures hold complex datatypes, for example, a table input to a SAP operation.

subscription – A subscription defines how data is transferred between a user's mobile device and Unwired Server. Subscriptions are used to notify a device user of data changes, then these updates are pushed to the user's mobile device.

Sybase Control Center – Sybase Control Center is the Flash-based Web application that includes a management framework for multiple Sybase server products, including Unwired Platform. Using the Unwired Platform administration perspective in Sybase Control Center, you can register clusters to manage Unwired Server, manage domains, security configurations, users, devices, connections, as well as monitor the environment. You can also deploy and MBO or workflow packages, as well as register applications and define templates for them. Only use the features and documentation for Unwired Platform. Default features and documentation in Sybase Control Center do not always apply to the Unwired Platform use case.

Sybase Control Center *X.X* Service – Provides runtime services to manage, monitor, and control distributed Sybase resources. The service must be running for Sybase Control Center to run. Previously called Sybase Unified Agent.

Sybase Hosted Relay Service – The Sybase Hosted Relay Service is a Web-hosted relay server that enables you to test your Unwired Platform development system.

Sybase Messaging Service – The synchronization service that facilitates communication with device client applications.

Sybase Unwired Platform – Sybase Unwired Platform is a development and administrative platform that enables you to mobilize your enterprise. With Unwired Platform, you can develop mobile business objects in the Unwired WorkSpace development environment, connect to structured and unstructured data sources, develop mobile applications, deploy mobile business objects and applications to Unwired Server, which manages messaging and data services between your data sources and your mobile devices.

Sybase Unwired WorkSpace – Sybase Unwired Platform includes Unwired WorkSpace, which is a development tool for creating mobile business objects and mobile applications.

synchronization – A synchronization method where data is delivered synchronously using an upload/download pattern. For push-enabled clients, synchronization uses a "poke-pull" model, where a notification is pushed to the device (poke), and the device fetches the content (pull), and is assumed that the device is not always connected to the network and can operate in a disconnected mode and still be productive. For clients that are not push-enabled, the default synchronization model is pull. *See also* messaging based synchronization.

synchronization group – Defined in Unwired WorkSpace, a synchronization group is a collection of MBOs that are synchronized at the same time.

synchronization parameter – A synchronization parameter is an MBO attribute used to filter and synchronize data between a mobile device and Unwired Server.

synchronization phase – For replication based synchronization packages, the phase can be an upload event (from device to the Unwired Server cache database) or download event (from the cache database to the device).

synchronize – *See also* data refresh. Synchronization is the process by which data consistency and population is achieved between remote disconnected clients and Unwired Server.

truststore – The location in which certificate authority (CA) signing certificates are stored. *See also* keystore.

undeploy – Running **undeploy** removes a domain package from an Unwired Server.

Unwired Server – The application server included with the Sybase Unwired Platform product that manages mobile applications, back-end EIS synchronization, communication, security, transactions, and scheduling.

user – Sybase Control Center displays the mobile-device users who are registered with the server.

view – A window in a perspective that displays information about one or more managed resources. Some views also let you interact with managed resources or with Sybase Control Center itself. For example, the Perspective Resources view lists all the resources managed by the current perspective. Other views allow you to configure alerts, view the topology of a replication environment, and graph performance statistics.

Visual Studio – Microsoft Visual Studio is an integrated development environment product that you can use to develop device applications from generated Unwired WorkSpace code.

Welcome page – In Eclipse, the first set of pages that display when you launch the application.

workspace – In Eclipse, a workspace is the directory on your local machine where Eclipse stores the projects that you create.

WorkSpace Navigator – In Eclipse, the tree view that displays your mobile application projects.

WSDL file – Web Service Definition Language file. The file that describes the Web service interface that allows clients to communicate with the Web service. When you create a Web service connection for a mobile business object, you enter the location of a WSDL file in the URL.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W