System Administration

Sybase Unwired Platform 2.2 SP04

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.

Check the Sybase Product Documentation Web site regularly for updates: *http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&amp;c=firsttab&amp;a=0&amp;p=categories* , 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 PreConfiguredUserLoginModule (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. In a deployment edition, you must 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).

Unwired Platform Security

Use Sybase Control Center to remotely and safely configure most of the security features for this product.

Security consists of:

- Securing server components and internal communications
- Securing the mobility environment and external communications with the platform runtime

Security is extensively documented in the *Security* guide.

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

CHAPTER 3 Cluster Administration

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 *Designing the Landscape* in *Landscape Design and Integration*.

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 setup 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.

Task	Frequency	Accomplished by
Configuring the cluster to: • Set the configuration cache properties • Set the solution manager URL • Set the replication ports and properties • Set the messaging synchronization ports • Set the amount of data that can be downloaded through an HTTP connection using the BlackBerry MDS • Set the management ports for communication requests from Sybase Control Center • Configure the client dispatcher • Set how applications handle DCN requests sent though an HTTP GET operation • Create security profiles for secure communication • Set up secure synchronization • Tune server performance	Post installation configuration with infrequent tuning as required	See *Sybase Control Center for Sybase Unwired Platform>Administer> Clusters*
Setting cluster log file settings for Unwired Server system components	Once, unless log data requirements change	See *Sybase Control Center for Sybase Unwired Platform>Administer>>Clusters*
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.

Task	Frequency	Accomplished by
Reviewing licensing information, including total licensed devices and currently used licenses count	Occasional, or as device user registration and deregistration occurs	Sybase Control Center.

CHAPTER 4 Configuring Asynchronous Operation Replay Queue Count

Configure properties of a cluster to control 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. 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.

CHAPTER 5 Configuring Cluster Performance Properties

To optimize Unwired Server performance across the cluster, configure the thread count and pool size, web service connection counts and timeout, inbound and outbound messaging queue counts, and proxy connection pool and synchronization cache size.

1. In the left navigation pane, select **Configuration**.
2. In the right administration pane, click the **General** tab.
3. From the menu bar, select **Performance**.
4. Configure these properties, as required:
    - Inbound messaging queue count – number of message queues used for incoming messages from the messaging-based synchronization application to the server. Sybase recommends you choose a value that represents at least 10% of active devices.
    - Maximum count of total webservice connections – maximum number of web service connections allowed overall.
    - Maximum count of webservice connections per host – maximum number of web service connections allowed per host configuration.
    - Maximum number of in memory messages – maximum allowable number of in memory messages.
    - Maximum proxy connection pool size – maximum size for the replication protocol server memory cache.
    - Outbound messaging queue count – number of message queues used for outbound messages from the server to the messaging-based synchronization application. Sybase recommends 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 – maximum number of threads allocated to initial bulk load subscription operations. The default value is 5 Setting the thread pool size too high can impact performance.
    - Synchronization cache size – maximum size for the replication protocol server memory cache.
    - Thread count – MobiLink thread count. This value should be lower than the thread count for the SQL Anywhere database.
    - Webservice connection timeout – length of time, in seconds, until a web service connection is established. A value of 0 (zero) means no timeout.
5. Click **Save**.

CHAPTER 6 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 for Sybase Unwired Platform*.

Configuring a Replication Listener

(Not applicable to Online Data Proxy)Configure the port to receive synchronization requests from client devices.

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 Server. 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, select **Configuration** .
3. In the right administration pane, click the **General** tab.
4. From the menu bar, select **Components**.
5. Select **Replication** and click **Properties**.
6. Select the protocol and port you require:

   - If you do not require SSL encryption, choose **Port**. Sybase Unwired Platform 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 can be used to review and set the server identity and public certificate for the secure synchronization port. See below.

7. (Optional) 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 – specify the file containing the private key that acts as the identity file for Unwired Server.
- E2E Encryption Certificate Password – set the password to unlock the encryption certificate.
- E2E Encryption Public Key – specify the file containing the public key 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.

Option	Description
`-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.
`-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.
`-zf`	(Recommended in development environments) Causes the Unwired Server replication service to check for script changes at the beginning of each synchronization. Unless this option is used, the service assumes that no script changes have been made, no checks for script changes are performed, once the service starts. For production environments, this option is not recommended due to its negative impact on synchronization performance.
`-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.

8. Click **OK**.

CHAPTER 7 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 for Sybase Unwired Platform*.

Configuring Messaging Properties

Configure a port to receive service requests from devices.

1. Open Sybase Control Center.
2. In the left navigation pane, select **Configuration**.
3. In the right administration pane, click the **General** tab.
4. From the menu bar, select **Components**.
5. Select **Messaging**, then click **Properties**.
6. Enter the synchronization port number. The default is 5001.
7. (Optional) Configure the BES response portioning value.

   There are limits on the amount of data that can be downloaded through an HTTP connection using the BlackBerry MDS. The BES response portioning limit determines the amount of HTTP traffic the BES MDS server accepts from Unwired Server. For BlackBerry MDS, this limit is set in BlackBerry Manager using the Maximum KB/ Connection setting.
8. Click **OK**.

CHAPTER 8 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.

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*.

CHAPTER 9 Server Administration

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.

There are two types of server nodes that can be installed:

- Application Server node – (mandatory) runs all services.
- Scale Out node – (optional) specifically designed to allow the stateless request/response HTTP and synchronous message services to be horizontally scaled.

Table 2. Server administration tasks

Task	Frequency	Accomplished by
Installing the server	One-time installation per server	Unwired Platform installer.
Tuning the server performance.	Post installation with infrequent tuning as required	Sybase Control Center. See *Sybase Control Center for Sybase Unwired Platform>Administer> Unwired Server*
Manage the outbound enabler configuration for Relay Server. • Configure Relay Server properties • Manage certificates • View logs • Configure proxy servers for outbound enabler	Post installation	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 cluster database:
   a) Using dbisqlc, connect to the cluster database.
   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. 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 Sybase Control Center. Use the new hostname instead of the old one in the launched URL
6. For a cluster environment, repeat step 2 for each node.
7. Restart the Sybase Unwired Server service.

Methods for Starting and Stopping Unwired Server

You can start and stop Unwired Server in different ways, 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.

Method	Use When	Services Stopped
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.

Note: Sybase Control Center requires at least one Application Server node running to work properly. If all Application Server nodes are stopped, the Sybase Control Center console tree will be collapsed automatically and the following error message will display: `The cluster is unavailable because of no running primary server found`.

Note: You cannot start a Scale-out node from Sybase Control Center. If you stop a Scale-out node, you must start it manually.

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

     Note: If you have selected a Scale-out node server type, the **Start** button is disabled.
   - 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.

Configuring Unwired Server Performance Properties

To optimize Unwired Server performance, configure the thread stack size, maximum and minimum heap sizes, and user options such as enabling trace upload to SAP® Solution Manager.

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. Configure these replication payload properties, as required:
    - Host Name – the name of the machine where Unwired Server is running (read only).
    - Maximum Heap Size – the maximum size of the JVM memory allocation pool. Use K to indicate kilobytes, M to indicate megabytes, or G to indicate gigabytes. For production recommendations on this value, see *Unwired Server Replication Tuning Reference* in *System Administration*.
    - 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*.
    - Thread Stack Size – the JVM `-Xss` option.
    - User Options (in Show optional properties) – 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. Finally, you can define the Solution Manager URL into which trace files can be uploaded for troubleshooting assistance: `-Dcom.sap.solutionmanager.url=<SolutionManager_URL>`.
6. Click **Save**.

See also

- *Tuning Synchronization for Messaging Payloads* on page 63
- *Testing CPU Loads for Replication* on page 54

Configuring Unwired Server to Prepare for Connections Outside the Firewall

Configure the Relay Server in Sybase Control Center to create the number of Relay Server farms for your clusters: at minimum, you should use one farm for replication and one for messaging synchronization payloads.

1. Install Relay Server on either an IIS or Apache Web server that host Relay Server as a Web proxy. See *Landscape Design and Integration* for Relay Server installation and configuration information.
2. Use Sybase Control Center to configure a Unwired Server cluster with farms, and nodes and their tokens, as needed, and with Relay Server connection information.
3. Distribute connection information to development teams so clients can connect using this information. See *Device Clients* in the *Security* guide.
4. Generate the Relay Server and Outbound Enabler configuration files from Sybase Control Center; distribute them as required. Ensure that farms for replication and messaging are configured accordingly. See *Relay Server* in *Sybase Control Center for Sybase Unwired Platform* online help.
5. Use a test application to ensure the connection is successful. If not, use logs, monitoring, and trace files to determine the source of the potential problem. See Developer Guides for you application type for information on application development.

Relay Server Components Used in Unwired Platform

Relay Server operations include several executable components.

These components include:

Relay Server host (rshost.exe) – the host resides on the Relay Server. It is responsible for accepting a single, inbound connection from the Unwired Server outbound enabler; accepting multiple, inbound connections from Afaria clients; and handling the associated processes that occur on the relay server for Unwired Server sessions. Install the relay server with files that are bundled with the Unwired Platform product. You define its configuration settings using Sybase Control Center.

Relay Server outbound enabler (rsoe.exe) – the outbound enabler is the relay agent on the Unwired Server. It is responsible for initiating an outbound connection with the relay server, while sustaining a connection with the Unwired Server .An outbound enabler is installed automatically when you configure it from Sybas.e Control Center.

Unwired Platform clients must be configured to use corresponding connection settings for Relay Server but do not require a separate, executable component.

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 for Sybase Unwired Platform*.

CHAPTER 10 Data Tier Administration

If you need to administer components that make up the data tier, review the tasks you can perform.

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 *Security* .

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 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 and monitor database (which share the same port).

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 and modify the `tcpip(PORT=newPort)` entry. 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. Make corresponding port number changes for eace datasource configuration files, which is located in *`UnwiredPlatform_InstallDir`*`\Servers\UnwiredServer\Repository\Instance\com\sybase\djc\sql\DataSource`.

For example, `default.properties` for default database, and `clusterdb.properties` for the cluster database.

4. Restart the cluster database.
5. 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.
6. Restart all Unwired Servers.

See also

- *Changing SQLAnywhere Database Server Thread Count and User Startup Options* on page 28
- *Update Properties (updateprops.bat) Utility* on page 236
- *Cache Database Startup Options (cdboptions.ini) Initialization File* on page 287

Changing SQLAnywhere Database Server Thread Count and User Startup Options

Change the database server startup options (thread counts and user 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 Unwired Servers.

   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 startup option change, open the corresponding initialization file. These files are installed to `SUP_HOME\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. Modify one of these properties: *.threadcount or *.useroptions in the `.ini` files, for example:

   ```
   -c 24M -gn 300
   ```
4. Ensure the cluster database is restarted.
5. 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
   ```
6. Restart all Unwired Servers.

See also

- *Changing Database Ports for SQLAnywhere Databases* on page 27
- *Update Properties (updateprops.bat) Utility* on page 236
- *Cache Database Startup Options (cdboptions.ini) Initialization File* on page 287

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 `SUP_HOME\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.

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 `SUP_HOME\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 for Sybase Unwired Platform*.
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 140

Data Tier Connection Pooling

Managing Connection Pools for Unwired Server Connections

Connection pools are used by Unwired Server to improve performance between the server and internal databases, which include cache database, cluster database, domainlog database, messaging database, monitor database and sampledb. These values are synchronized among all servers in the cluster when the primary server values change.

Configure the maximum pool size to determine the how large of a pool is used for these JDBC database connections.

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. The Max Pool Size value should not be a negative value.

   See *Creating Connections and Connection Templates* in *Sybase Control Center for Sybase Unwired Platform* and *JDBC Properties*.
7. Click **Save** to save the changes.

See also

- *JDBC Properties* on page 193

Rebuilding 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 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 *Backup and Recovery*.
3. Rebuild each runtime database with the SQL Anywhere Unload (dbunload) utility.

   The dbunload utility is available in the `*SUP_HOME*\Servers\SQLAnywhere12\BIN32` directory. For details, see *http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.sqlanywhere.12.0.1/dbadmin/dbunload.html* .
4. Validate the data before restarting the services.

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;
```

CHAPTER 11 EIS Connection Management

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 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 with the Connections node
Create and maintain connection pool templates	One-time	Sybase Control Center with the Connections node
Update package connections when moving from development and test environments to production environments	On-demand, as needed	Sybase Control Center with the **Deployment Wizard**

See also

- *EIS Data Source Connection Properties Reference* on page 193
- *Creating Data Source Connections for a Domain* on page 44

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.

See also

- *JCo Connection Rules* on page 37

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 193

Tuning Connection Pool Sizes

Platform administrators can tune the maximum pool size to data source connections for improved performance. This ensure that adequate EIS resources are allocated for servicing the Unwired Platform runtime.

1. In the navigation pane of Sybase Control Center, expand the domain you want to tune pool sizes for, then click **Connections**.
2. In the administration pane, select the EIS connection, then click **Properties**.
3. Change the **Max pool size** property to `100`.
4. Click **Test Connection** and make sure the server can still be pinged.
5. If you can connect to the server, click **Save**.

See also

- *EIS Connection Performance Tuning Reference* on page 57

JCo Connection Rules

Understand default behavior when configuring Unwired Server to SAP enterprise information system JCo connections.

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

1. Each distinct connection of credentials and connection parameters creates a new destination name.

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

See also

- *Data Source Connections* on page 35

CHAPTER 12 Domain Management

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 cluster.

- An 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 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. Create a security configuration, then map it 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*

   Map logical roles to physical roles for a domain by setting the mapping state. Domain administrators map roles for the domains they control. Role mappings performed at the domain level are automatically applied to all domains that share the same security configuration. Domain-level role mapping overrides mapping set at the cluster level by the platform administrator.
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 primarily 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**.

   Note: Domain names are case insensitive.
5. Select a security configuration for the domain by checking an option from the list of available configurations. You must select at least one security configuration. The security configurations you select are then available for use in validating users accessing the packages. If you select multiple security configurations, the first one you select becomes the default security configuration for the domain.
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. Create a security configuration, then map it 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**, **Attribution**, 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 for Sybase Unwired Platform*.
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.

Mapping Roles for a Domain

Map logical roles to physical roles for a domain by setting the mapping state. Domain administrators map roles for the domains they control. Role mappings performed at the domain level are automatically applied to all domains that share the same security configuration. Domain-level role mapping overrides mapping set at the cluster level by the platform administrator.

Prerequisites

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

Task

1. In the left navigation pane of Sybase Control Center, expand **Domains** > ***Domain name***. > **Security** and select the security configuration to map roles for.
2. In the right administration pane, click the **Role Mappings** tab.
3. Select a logical role and select one of the following in the adjacent list:

State	Description
AUTO	To map the logical role to a physical role of the same name.
NONE	To disable the logical role, which means that the logical role is not authorized.
MAP	To manually map the logical role when the physical and logical role names do not match. See *Mapping a Physical Role Manually*.

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 *Connections* in *Sybase Control Center for Sybase Unwired Platform* and *EIS Connection Management*.

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

- *Chapter 11, EIS Connection Management* on page 35

Scheduling Accumulated Data Cleanup for Domains

Periodically clean up accumulated data maintenance items in cache that are no longer needed, by creating a data purge schedule. The schedule should be set to perform the clean up processes run when system usage is low.

However, you can also manually purge accumulated data items at any time; however note that for domain logs, if the number of domain log data is very large (one with hundreds of thousands of entries for example) then Unwired Server must purge domain log data asynchronously.

Note: You can use the Administration API to automate clean up at the package level.

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 not active for the 'number of inactive days' in the schedule task configuration. Note, subscription is considered active as follows: • Replication – last synchronization request time-stamp. • Messaging – last synchronization message time-stamp. **Note:** If a casual user accesses the system infrequently, for example three to four times a year, the user falls outside the specified time frame and is removed from Sybase Unwired Platform. The user will then have to reinstall Sybase Unwired Platform and initiate a sync to re-activate subscription.
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	This cleanup task 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. • Unused or stale partitions. • Expired DCN entries identified by the **exp** property in the DCN message.

4. Select **Enable**. Schedules run until you disable them, or they expire.

CHAPTER 13 Performance Tuning and Testing

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. Load testing measures performance under simulated use.

Tuning recommendations vary depending the payload protocol.

Performance Considerations for Relay Server and Outbound Enabler

Use this table to quickly ascertain which Relay Server and Outbound Enabler configuration properties you can adjust to tune performance.

Table 4. Recommendations 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
Relay Server response times	To increase the response time of a slow Relay Server, add `up_pad_size=0` to the Relay Server configuration file created from Sybase Control Center.	None	0

Component	Function	Default	Production Recommendation
Outbound Enabler 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

See also

- *Performance Considerations for Replication* on page 48
- *Performance Considerations for Messaging* on page 58
- *LoadRunner Extension for Sybase Unwired Platform* on page 65
- *Enabling RBS Performance Testing with LoadRunner* on page 71

Performance Considerations for Replication

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 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 a 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 Relay Server and Outbound Enabler* on page 47
- *Performance Considerations for Messaging* on page 58
- *LoadRunner Extension for Sybase Unwired Platform* on page 65
- *Enabling RBS Performance Testing with LoadRunner* on page 71

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.

Unwired Server Replication Tuning Reference

Use this table to quickly ascertain which runtime properties you can adjust to tune replication performance.

Table 5. 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.
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.	70p where p is is a percentage either of the physical system memory, or of the process addressable space, whichever is lower.	On 64-bit database servers, the cache size can be considered unlimited .
JVM minimum heap size	The minimum memory allocated to the differencing and cache management functions of the server.	512MB	

Property	Function	Default	Production Recommendation
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.	2048MB	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

Tuning Synchronization for Replication Payloads

If your applications synchronize over the replication payload protocol, 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 for Sybase Unwired Platform* 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 new cache thread count, pool size, and initial cache size values.

   See *Changing SQLAnywhere Database Server Thread Count and User Startup Options* and *Managing Connection Pools for Unwired Server Connections* .
7. Restart the cache service and all Unwired Servers.
8. In Sybase Control Center, update the Unwired Server replication synchronization properties on the cluster.

   a) In the left navigation pane, click **Configuration**.
   b) In the right administration pane, click the **General** tab.
   c) To configure replication ports and optional properties, select **Components** from the menu bar, select **Replication**, and click **Properties**.
   d) To configure **Synchronization Cache Size** and **Thread Count**, select **Performance** from the menu bar, For production recommendations on the values for these properties, see *Unwired Server Tuning Reference* in *System Administration*.
9. 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.
   c) Adjust the thread stack size and JVM heap sizes to the new values.
      - 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. Use K to indicate kilobytes, M to indicate megabytes, or G to indicate gigabytes. For

production recommendations on this value, see *Unwired Server Replication Tuning Reference* in *System Administration*.
- User Options (in Show optional properties) – 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.

d) Repeat for all servers.

10. 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.

See also

- *Configuring Unwired Server Performance Properties* on page 24

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 6. 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)

Configuring Initial Cache Size on a Single Node for Performance

Configure the initial memory reserved for caching database pages on a single node installation to enhance performance.

1. Stop Unwired Server.

   See *Methods for Starting and Stopping Unwired Server.*
2. Shutdown and remove the consolidated database (CDB) service using the batch file in *SUP_HOME*`\Servers\UnwiredServer\bin`.

   ```
   asadbservice.bat stop
   ```

   ```
   asadbservice.bat remove
   ```
3. Set Unwired Platform services to manual start if they are designated as automatic start.
4. Restart Unwired Server.
5. Modify *SUP_HOME*`\Servers\UnwiredServer\config\configure-sup.xml` as follows:

   Change `-c 24M` to `-c 8G`.
6. Reinstall the CDB service.

   ```
   asadbservice.bat install manual
   ```

   or

```
asadbservice.bat install auto
```

7. Set Unwired Platform services to automatic start.
8. Restart Unwired Server.

Configuring Initial Cache Size on a Cluster for Performance

Configure the initial memory reserved for caching database pages on a cluster installation to enhance performance.

1. Stop Unwired Server in each node.

   See *Methods for Starting and Stopping Unwired Server.*
2. Stop the Cache DB service.
3. Modify `SUP_HOME\Servers\SQLAnywhere16\BIN64\cdboptions.ini` as follows:

   Change `-c 24M` to `-c 8G`.
4. Start the Cache DB service.
5. Start Unwired Server in each node.

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)

See also

- *Tuning Connection Pool Sizes* on page 37

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 Relay Server and Outbound Enabler* on page 47
- *Performance Considerations for Replication* on page 48
- *LoadRunner Extension for Sybase Unwired Platform* on page 65
- *Enabling RBS Performance Testing with LoadRunner* on page 71

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.

Performance Considerations for Relay Server and Outbound Enabler

Use this table to quickly ascertain which Relay Server and Outbound Enabler configuration properties you can adjust to tune performance.

Table 7. Recommendations 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
Relay Server response times	To increase the response time of a slow Relay Server, add `up_pad_size=0` to the Relay Server configuration file created from Sybase Control Center.	None	0
Outbound Enabler 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 8. 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
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

Property	Function	Default	Production Recommendation
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
JVM maximum heap size	The maximum memory allocated to the differencing and cache management functions of the server.	2GB	6GB

* Cluster-affecting property

Messaging Maximum Content Sizes

To support messaging performance, beyond the properties available in Sybase Control Center, you can configure the *sup.msg.max_content_size* property in the cluster database. This property is only available in the database; modify the default only after discussing the correct value with a Sybase representative.

To change *sup.msg.max_content_size*, by running the following SQL command using the method of your choosing:

```
update cluster_prop set value='newValue' where
name='sup.msg.max_content_size'
```

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 message size exceeds the limit, the message is immediately sent to the client side.

Tuning Synchronization for Messaging Payloads

If your applications synchronize over the messaging payload protocol, tune your production environment after all components have been deployed and started.

1. Isolate the monitoring database from the cache server.

   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 *Sybase Control Center for Sybase Unwired Platform*.
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 on the cluster.

   a) In the left navigation pane, click **Configuration**.
   b) In the right administration pane, click the **General** tab.
   c) Select **Performance** from the menu bar.
   d) Change the default values for inbound and outbound queues to 100 each.
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 MBO Packages* and *MBO Package Management* in *Sybase Control Center for Sybase Unwired Platform*.

   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).

See also

- *Configuring Unwired Server Performance Properties* on page 24

LoadRunner Extension for Sybase Unwired Platform

HP LoadRunner is an automated performance and testing software used to examine system behavior and performance under actual load conditions. Use the Sybase Unwired Platform extension to integrate with this product.

Note: The LoadRunner Extention for Sybase Unwired Platform applies only to Hybrid Web Container applications and any applications that use the Online Data Proxy through the messaging channel. For information on using LoadRunner without the extension, see *Enabling RBS Performance Testing with LoadRunner*.

The LoadRunner Extension is installed in `SUP_HOME\Servers\LoadRunnerAPI`. However, Unwired Platform does not install LoadRunner: you must install a compatible version of LoadRunner separately, as specified in *Supported LoadRunner Versions* in *Supported Hardware and Software*. The Extension is designed to be used by experienced LoadRunner testers.

The Extension has two components:

Code generator – reads an XML recording file that you create using the Unwired Server, and generates C# script files that LoadRunner can execute.

Runtime library – provides APIs, referenced by the generated C# script files, that LoadRunner uses in addition to its own APIs.

See also

- *Performance Considerations for Relay Server and Outbound Enabler* on page 47
- *Performance Considerations for Replication* on page 48
- *Performance Considerations for Messaging* on page 58
- *Enabling RBS Performance Testing with LoadRunner* on page 71

Enabling Performance Testing with LoadRunner

Use LoadRunner to load test a Hybrid Web Container or Open Data Protocol client application.

1. *Enabling and Disabling XML Trace Recording*

   Control when the Unwired Server records communication between the client and the server.
2. *Recording XML Trace Files*

   Generate an XML file that records each communication between client and server.
3. *Generating LoadRunner Extension Script Files*

   Use the extension to create C# script files for LoadRunner.

4. *Parameterizing Generated Scripts*

   Modify the generated scripts to support access by multiple users.
5. *Compiling Extension Scripts and Configuring LoadRunner*

   Create compiled LoadRunner files from your parameterized scripts, and configure LoadRunner to run the scripts.
6. *Configuring Security*

   Configure your application connection template to handle simulated test users.
7. *Registering HWC Application Connections*

   Assign the HWC application to a connection template.

Enabling and Disabling XML Trace Recording

Control when the Unwired Server records communication between the client and the server.

Prerequisites

If you have a cluster environment, shut down all but one member of your cluster before recording. LoadRunner extension recording is designed to work on a single server.

- To enable recording, run `SUP_HOME\Servers\UnwiredServer\LoadRunnerAPI\bin\start-recording.exe`
  The next time you run your client application, the server records Mobile Objects protocol communication between the client and server.
- To disable recording, run `SUP_HOME\Servers\UnwiredServer\LoadRunnerAPI\bin\stop-recording.exe`
  Disable XML recording before running a LoadRunner test, because the overhead associated with recording might affect server performance.

Next

If you shut down or suspended servers in a cluster before recording, you can restore the servers after the recording.

Recording XML Trace Files

Generate an XML file that records each communication between client and server.

To generate XML trace files after enabling trace recording on the server, run any HWC or ODP client application on your favorite device (for example, the Sybase Hybrid App on an iPad or Android device).
To generate a trace file, simply run the client application normally.
Each interaction between the client and the server is recorded in an XML recording file at the server. Each active application connection generates a separate file.

The XML recording files are written in `SUP_HOME\Servers\UnwiredServer\logs\XMLRecording XML`.

You can identify the recording files for HWC applications by the suffix "__HWC.xml". To identify recordings that apply to ODP applications, search recording files for the string "GWCRequest".

Notes:

- You can safely delete an XML recording file if its associated client is inactive. This enables you to start a new recording for that client, ignoring any previous requests and responses.
- Some additional generated files are associated with HWC application deployment or administration. If you find an XML recording file with a name that includes the string "LR*n*", where *n* >= 1, it is likely you have left XML recording enabled while running a LoadRunner test. Recording during a benchmark test can adversely impact server performance.

Generating LoadRunner Extension Script Files

Use the extension to create C# script files for LoadRunner.

At a command prompt, run one of these commands in `SUP_HOME\Servers\UnwiredServer\LoadRunnerAPI\bin` to generate LoadRunner scripts (where *trace-file* is the full path of an XML recording file):

- **For HWC –** `vugen-hwc "trace-file"`
- **For ODP –** `vugen-odp "trace-file"`

By default, the commands generate script files in one of these output subdirectories under `SUP_HOME\Servers\UnwiredServer\genfiles\cs\src`:

```
VuGenHwc\
VuGenOdp\
```

To specify an alternate output directory, use the -od option:

```
vugen-hwc "my-trace-file.xml" -od "my-output-dir"
```

The command generates these C# script files (after creating the output directory if necessary):

- `vuser_init.cs` – initialization code.
- `Action.cs` – main script containing most of the functional LoadRunner code.
- `vuser_end.cs` – cleanup code.

Parameterizing Generated Scripts

Modify the generated scripts to support access by multiple users.

The C# code generated by the LoadRunner extension provides basic load-testing functions based on the trace files for your application run. However, as generated, the code is unsuitable for realistic, multiuser load testing. For example, a script may try to update an entity with a particular enterprise information system (EIS) primary key that was seen in the recording. When you run the test from LoadRunner Controller with multiple virtual users, there might be simultaneous attempts to create, update, or delete the same EIS data object, which might cause EIS operation failures.

Therefore, you must modify the generated code using LoadRunner parameterization techniques. Usually, this means changing constant strings into variable references or other expressions.

In addition to the LoadRunner API documentation, see the API documentation for the Sybase Unwired Platform LoadRunner extension, which is installed in `SUP_HOME\Servers\LoadRunnerAPI\docs\api`.

For example, consider this generated Action method:

```
public int Action()
{
  lr.debug_message(lr.MSG_CLASS_BRIEF_LOG, "Action1");
  key_Customer_create_address = "A";
  key_Customer_create_city = "C";
  key_Customer_create_company_name = "Sybase";
  key_Customer_create_fname = "F";
  key_Customer_create_id = "10001";
  key_Customer_create_lname = "L";
  key_Customer_create_phone = "1234567890";
  key_Customer_create_state = "S";
  key_Customer_create_zip = "12345";
  tx = "Action1_Customercreate_Online_Request";
  lr.start_transaction(tx);
  try
  {
    SAP.Mobile.LoadRunner.HwcMessage m1 = hwc.OnlineRequest
    (
      hwc.Screen("Customercreate"),
      hwc.Action("Online_Request"),
      hwc.Values
      (
        hwc.Value("Customer_create_id_paramKey", hwc.NUMBER, _
          key_Customer_create_id),
        hwc.Value("Customer_create_fname_paramKey", hwc.STRING,  _
          key_Customer_create_fname),
        hwc.Value("Customer_create_lname_paramKey", hwc.STRING,  _
          key_Customer_create_lname),
        hwc.Value("Customer_create_address_paramKey", hwc.STRING,  _
          key_Customer_create_address),
        hwc.Value("Customer_create_city_paramKey", hwc.STRING,  _
          key_Customer_create_city),
        hwc.Value("Customer_create_state_paramKey", hwc.STRING,  _
          key_Customer_create_state),
        hwc.Value("Customer_create_zip_paramKey", hwc.STRING,  _
          key_Customer_create_zip),
        hwc.Value("Customer_create_phone_paramKey", hwc.STRING,  _
          key_Customer_create_phone),
        hwc.Value("Customer_create_company_name_paramKey",  _
          hwc.STRING, key_Customer_create_company_name),
        hwc.Value("ErrorLogs", hwc.LIST)
      )
```

```
   );
   hwc.ExpectScreen(m1, "Customercreate");
   lr.end_transaction(tx, lr.PASS);
 }
 catch (System.Exception _ex_1)
 {
   lr.error_message(_ex_1.ToString());
   lr.end_transaction(tx, lr.FAIL);
 }

 return 0;
}
```

To avoid creating the same name for all customers, you might change the “Sybase” constant to an expression. For example:

```
key_Customer_create_company_name = "Company" +
Math.Abs(lr.vuser_id);
```

Compiling Extension Scripts and Configuring LoadRunner

Create compiled LoadRunner files from your parameterized scripts, and configure LoadRunner to run the scripts.

1. Create empty C# script files, using these parameters:
   a) Configure the Virtual User Generator to use the Microsoft .NET protocol and the default C# language option.
   b) Do not start recording: you will use the extension-generated recording instead.

   In the output folder, LoadRunner generates stub C# source files for `vuser_init.cs`, `Action.cs`, and `vuser_end.cs`. It also generates a C# project file, `Script.csproj`.
2. From your extension output folder, copy the C# script files that you generated using the extension, and paste them into the location of the empty LoadRunner-generated files that you generated in the first step.

   This overwrites the empty stub files.
3. Edit the copied extension script files as required, including your parameterization.
4. If you are redeploying a Hybrid App, verify that the generated ID and version are still correct.

   The generated `vuser_init.cs` includes the ID and version from the XML recording. For example:

   ```
   hwc.ModuleId = 1;
   hwc.ModuleVersion = 2;
   ```

   When you deploy the Hybrid App, the server assigns a module ID and version number, which at runtime should match the values in the generated file. However, the server might not assign the same values each time, even if you have not changed the application.

a) In `SUP_HOME\Servers\MessagingServer\Bin\Mobile Workflow`, note the ID and version number for the redeployed Hybrid App.
b) Compare these values to the ID and version number in the `vuser_init.cs` file. If the values are different, correct the values in `vuser_init.cs`.

5. In the LoadRunner-generated C# project, add a reference to `SUP_HOME\Servers\UnwiredServer\LoadRunnerAPI\lib\SAP.Mobile.LoadRunner.dll`. A convenient method is to open `Script.csproj` in Visual Studio, add the reference there, then save the project and exit Visual Studio.

Note: If you generate `Script.csproj` using LoadRunner 11.0, then open `Script.csproj` using Visual Studio 2008 or later, Visual Studio upgrades your C# project. The LoadRunner 11.0 Virtual User Generator and Controller cannot build the upgraded project, because LoadRunner invokes an incompatible version (2.0) of the msbuild tool.

To work around this problem, edit the `Script.sln` file, and change the Format Version from 10.00 to 9.00. The problem does not occur if you use LoadRunner 11.5.

6. Compile the script in Virtual User Generator.

7. In Virtual User Generator, open Run-time Settings. In the Miscellaneous section under Multithreading, choose **Run Vuser as a process**.

The default Run Vuser as a process option is not supported.

8. Copy the `DLL` files from `SUP_HOME\MobileSDKversion\Win32` to the `bin` subfolder of your LoadRunner script directory.

These DLLs are required at runtime to execute LoadRunner extension scripts.

Configuring Security

Configure your application connection template to handle simulated test users.

By default, the LoadRunner extension scripts assume a range of users from `user000001` to `user999999` in the test security configuration. The numeric suffix is based on the LoadRunner `vuser_id`. If you retain this credential generation scheme, configure your application connection template to use the test security configuration. Here are some configuration alternatives:

- In Sybase Control Center, select the NoSecLoginModule Security option in the Authentication tab. This is the simplest option for testing. Because the user name and password credentials are passed through to EIS, you should also configure the EIS to accept these credentials without checking the password, or alter the generated `vuser_init.cs` script to select an appropriate password for each user.
- Modify the generated `vuser_init.cs` script to choose a different credential generation scheme or security configuration name. If you use this method, also configure your application and security appropriately.

- Explicitly set the EisUsername and EisPassword properties, if they need to be distinct from the user name and password properties.

Registering HWC Application Connections

Assign the HWC application to a connection template.

Before testing an HWC application with the LoadRunner extension, use Sybase Control Center to assign the Hybrid App to the application connection template.

At runtime, LoadRunner automatically registers the device users (for example, user000001@test, user000002@test, and so on), enabling test users to access the Hybrid App.

No assignment is required to test ODP applications.

Enabling RBS Performance Testing with LoadRunner

Use the C# Mini-App method to perform load testing of RBS in Sybase Unwired Platform with LoadRunner.

1. Call `Sybase.Mobile.Application.GetInstance(USER_IDENTIFIER, USER_DATA_DIRECTORY);` to distinguish multiple clients on a single host.
2. Call `ConnectionProfile .SetProperty("databaseFile", USER_DATA_DIRECTORY + "\\database.udb");` to separate the database files for each client.
3. Configure `DatabaseFile` before calling `SampleDB.SetApplication(app);`
4. `Use ConnectionProfile.save()` to avoid contention.
5. Use `app.UnregisterApplication(TIMEOUT);` to block the invocation before `DeleteDatabase` for a successful cleanup.

See also

- *Performance Considerations for Relay Server and Outbound Enabler* on page 47
- *Performance Considerations for Replication* on page 48
- *Performance Considerations for Messaging* on page 58
- *LoadRunner Extension for Sybase Unwired Platform* on page 65

CHAPTER 14 Backup and Recovery

Sybase recommends that for all host nodes, use your standard complete backup procedures. That way, in case of catastrophic failure, you can completely restore these hosts to its previous state as of the last backup.

Complete backup of a host includes:

- File artifacts of installed software components that make up Sybase Unwired Platform.
- Registry entries (so that the installed software and services can all run properly after the restore).

Note: When backing up the Sybase Unwired Platform Runtime installation, the path to the Embedded Web Container in the `SCC-3_2` folder is too long for Windows to process. Before the back up, you must delete the contents of `*SUP_HOME*\services\EmbeddedWebContainer\container\Jetty-7.6.2.v20120308\work`. You must delete the contents of this folder from the command prompt, and not from Windows Explorer.

If you are only backing up a single node cluster, Sybase recommends that you reinstall to a new host and add a new node to existing cluster. This installation option automatically synchronizes the cluster ensuring the same messaging server keys and required artifacts are shared and configured correctly. (For example, RSOEs, installation of third-party software (for example, JDBC drivers for non-Sybase EIS databases). Always document these backup and configuration requirements in their internal project documentation. For assistance, see *Completing Installation Worksheets* in the *Landscape Design and Integration* guide.

Backup and Recovery of the Unwired Platform Data Tier

All runtime data is stored in the data tier. Therefore, you can backup the entire system relatively easily, and recover data as required. The recovery and post-recovery process you employ varies depending on the failure scenario you are addressing. Always diagnose your failure before following a particular course of action.

Exclusions: This documentation does not address system restoration or disaster recovery. For disaster recovery, you may need to work with third-party vendors who provide the necessary solutions. Furthermore, complete restoration (that is, the return of the system to its original runtime state before the failure occurred) cannot be guaranteed in any mobile environment.

The backup and recovery life cycle has having multiple stages.

Table 9. Backup and Recovery: Stages and Activities

Stage	Activity	Frequency
Planning	Identify the artifacts to be backed up and set the backup schedule.	Infrequent. Typically, before installing, or as business requirements or policies dictate.
	Test responses to all failures, including media failure, user error, application error, and component failure.	Occasional, as the deployment environment or scale changes.
Implementation	Configure backup in installed components, primarily data tier components.	One-time, or as change in frequency requires.
	Secure backup artifacts.	One-time, or as backup environment or number of artifacts change.
Maintenance	Monitor the backup environment.	Routine.
	Test the backup to ensure it is viable.	Routine.
Evaluation	Assess the issue and determine if recovery is required.	As needed.
Recovery	Recover from data loss or failure.	As needed.
Postrecovery	End user follow up activities to return it to its production-ready state.	As needed
Troubleshooting	Resolve any issues relating to the backup and recovery processes.	As needed.

Planning Backup and Recovery

Backup and recovery consists of activities that require coordination among the platform administrator, the security administrator, and IT. Planning ensures that you have all the information that you need before you implement a backup strategy.

1. *Understanding the Installation Environment*

   Learn where components are initially installed. When building connection strings for these databases, use the documented syntax.
2. *Determining the Backup Frequency and Location*

   The backup frequency of Sybase Unwired Platform is determined by a variety of factors.

See also

- *Backing Up Runtime Data with a Recovery Server* on page 77
- *Maintaining Backups* on page 82
- *Diagnosing and Recovering from Failure by Scenario* on page 83
- *Performing Postrecovery Activities* on page 85

Understanding the Installation Environment

Learn where components are initially installed. When building connection strings for these databases, use the documented syntax.

See also

- *Determining the Backup Frequency and Location* on page 77

SQL Anywhere Storage

The Unwired Platform installation creates runtime databases that jointly make up the data tier.

By default, these databases use transaction logs. Transaction logs are stored with the database file instances in the same folder or different folders, and share the same file name but use the `.log` extension. Transaction log mirroring is not enabled. The default locations of these database files and transaction logs is `*UnwiredPlatform_InstallDir*\Servers\UnwiredServer\data`:

- Cache database files: `default.db`, `default.log`
- Cluster database files: `clusterdb.db`, `clusterdb.log`
- Monitoring database files: `monitordb.db`, `monitordb.log`
- Domain log database files: `domainlogdb.db`, `domainlogdb.log`

In addition, samples and tutorials use `sampledb.db`, `sampledb.log`, but backing up these database files is entirely at your discretion.

Because all of these databases make up the data tier of the platform, backup activities for this tier requires the most attention and resources. Key recommendations from Sybase include:

- Storing the database file, transaction log, and the log mirror on different physical disks.
- Regularly rolling over the transaction log to minimize impact in case of storage failure.

Building Connections Strings for Unwired Platform

Many backup and recovery task require that you connect to a database using various command line tools. These tools use a connection string that defines connection variable values required for a given database.

Use the guidelines in the table to construct your connection string, and follow the recommendations in this topic. Before entering a connection string, use **dbisql** to verify that you can connect to the database with it. You can find **dbisql** in the `<UnwiredPlatform_Installdir>\Servers\SQLAnywhere12\BINXX` folder.

1. Use this syntax to construct a connection string:

```
dbn=DBNAME;uid=DBUSER;pwd=DBPWD;links=tcpip(DoBroadcast=NO;Verify
ServerName=NO;host=DBHOST;port=DBPORT)
```

Variable	Value
DBNAME	• **For the cache database –** use `default`. • **For the cluster database –** use `clusterdb`. • **For the monitoring database –** use `monitordb`. • **For the domain database –** use `domainlogdb`.
DBUSER	The name of database administrator user (by default, `dba`) .
DBPWD	The password of the database administrator user (by default, `sql`).
DBHOST	The host name or IP address of the data tier computer.
DBPORT	The port number of the database. The following list identifies default database ports. Ports may have changed; use the appropriate port values for your production environment. • **For a single-node complete data tier installation –** `5200`. • **For the cache database (cluster) –** `5200`. • **For the cluster database (cluster) –** `5300`. • **For the monitoring database (cluster) –** `5400`. • **For the domain database (cluster) –** `5400`.

2. Test the connection:
   ```
   dbisql -c connection-string
   ```
   For example, you can access the cache on the data tier installed in a cluster environment with:

   dbisql -c "dbn=default;uid=dba;pwd=sql;links=tcpip(DoBroadcast=NO;VerifyServerName=NO;host=localhost;port=5200)"

If you see a text entry window for entering SQL commands, then the connection string is valid; you can exit the program and use the connection string in all required command line utilities. A window that prompts for more connection parameters indicates that the connection string is invalid. Review the syntax and try again.

Determining the Backup Frequency and Location

The backup frequency of Sybase Unwired Platform is determined by a variety of factors.

1. Determine how frequently the backups need to occur. This is determined by the:
   - Volume of transactions
   - Use of transaction log mirroring. If you are not using mirroring, determine amount of data that can potentially be lost during the backup increment you select. A typical incremental backup frequency is anywhere from 5 to 60 minutes.
2. For each database and log, determine where the backups are saved and record that decision.

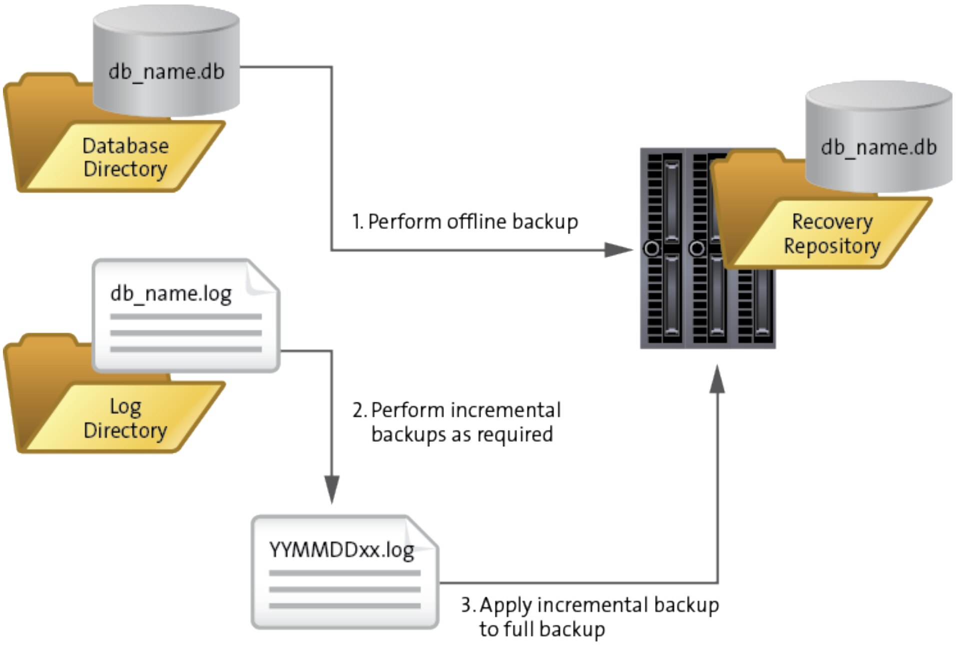
   The best practice is to backup to an entirely different physical location. For a list of database and log files you can back up, see *SQL Anywhere Storage*.

See also

- *Understanding the Installation Environment* on page 75
- *SQL Anywhere Storage* on page 75

Backing Up Runtime Data with a Recovery Server

Sybase recommends that you follow the documented steps to create a backup from which you can recover data.


db_name.db
Database Directory
1. Perform offline backup
db_name.db
Recovery Repository
db_name.log
Log Directory
2. Perform incremental backups as required
YYMMDDxx.log
3. Apply incremental backup to full backup

1. *Enabling Transaction Log Mirroring*

   (Optional) Enable transaction log mirroring, if your backup strategy requires it. Mirror the transaction log files using the dblog command line utility to store the primary and mirror transaction logs on separate drives from the database file.
2. *Creating a Recovery Server*

   Use a recovery server to hold all recovery data for all databases on the data tier node. Use this data to recover any database on the data tier in the event of data loss or corruption.
3. *Performing a Full Offline Backup After Install or Upgrade*

   Back up the entire Unwired Platform runtime and data tier. The recommended time to do this is immediately after installaing or upgrading and before you startup these services, as you cannot perform offline backups while data services are running. .
4. *Performing Periodic Incremental Transaction Log Backups Online*

   Use periodic incremental live backups to prevent transaction logs from growing too large. Databases can remain online while this type of backup is generated. You can then apply incremental backups against the full backup on the recovery server.
5. *Applying Increments to Full Backups on the Recovery Server*

   Update the full backup with contents of the transaction log.

See also

- *Planning Backup and Recovery* on page 74
- *Maintaining Backups* on page 82
- *Diagnosing and Recovering from Failure by Scenario* on page 83
- *Performing Postrecovery Activities* on page 85

Enabling Transaction Log Mirroring

(Optional) Enable transaction log mirroring, if your backup strategy requires it. Mirror the transaction log files using the **dblog** command line utility to store the primary and mirror transaction logs on separate drives from the database file.

1. Identify the target destination for transaction logs.

   The drive letter cannot be to a substituted drive.
2. From a command prompt, run:

   ```
   dblog -t c:\transaction_file.log -m m:\mirror_file.log db_name.db
   ```

See also

- *Creating a Recovery Server* on page 79

Transaction Log Mirrors

A transaction log mirror is an identical copy of the transaction log. If a database has a transaction log mirror, every database change is written to both the transaction log and the transaction log mirror.

Sybase recommends that you enable transaction log mirror for extra protection of high-volume or critical application data. It enables complete data recovery if a media failure on the transaction log occurs.

By default, databases on the data tier do not have transaction log mirrors enabled. Mirroring might degrade performance, depending on the nature and volume of database traffic and on the physical configuration of the database and logs.

To avoid losing both the log and the database file in the event of failure, Sybase suggests that you keep the mirror on a different physical disk.

Creating a Recovery Server

Use a recovery server to hold all recovery data for all databases on the data tier node. Use this data to recover any database on the data tier in the event of data loss or corruption.

Ensure that the recovery server is set up on a host node that is separate from the rest of Unwired Platform production nodes.

1. Use the Unwired Platform installer to install only data tier components on a separate host.
2. Do not install or use logs on the recovery host.

See also

- *Enabling Transaction Log Mirroring* on page 78

Performing a Full Offline Backup After Install or Upgrade

Back up the entire Unwired Platform runtime and data tier. The recommended time to do this is immediately after installaing or upgrading and before you startup these services, as you cannot perform offline backups while data services are running. .
Create a baseline for the recovery server. From a command prompt, run this command to copy the database file to a target directory:

```
copy db_name.db target-directory
```

See also

- *Performing Periodic Incremental Transaction Log Backups Online* on page 80
- *Unwired Platform Windows Services* on page 191

Securing Backup Artifacts

If you perform backups of Unwired Platform, you should also secure the backup artifacts.

1. Protect backups with Administrator and SYSTEM permissions.
2. Ensure that role-based access to the backup folder uses the same permissions model used for the production servers. The same IT users that can access the production database folders should be the same ones that can accessing the backup folders.
3. Perform any additional enterprise security requirements.

Performing Periodic Incremental Transaction Log Backups Online

Use periodic incremental live backups to prevent transaction logs from growing too large. Databases can remain online while this type of backup is generated. You can then apply incremental backups against the full backup on the recovery server.

Note: Incremental backups must be simultaneously indexed (renamed and a new output file started), particularly for the cache database. This prevents open-ended growth of the transaction log, while maintaining information about the old transactions.

1. *Using dbbackup to Create Incremental Backups*

   Use the dbbackup utility to create an incremental live backup that also resets transaction logs on completion. Each backup uses a unique name that acts as a method of versioning each file.
2. *Using dbtran to Validate the Backup*

   Validation allows you to check the integrity of incremental backups and to ensure that you can safely apply these increments to the offline backup. This is a vital step and cannot be bypassed, without risking the viability and integrity of your backup data.

See also

- *Performing a Full Offline Backup After Install or Upgrade* on page 79
- *Applying Increments to Full Backups on the Recovery Server* on page 81

Using dbbackup to Create Incremental Backups

Use the **dbbackup** utility to create an incremental live backup that also resets transaction logs on completion. Each backup uses a unique name that acts as a method of versioning each file.

dbbackup is in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere12\BINXX`.

1. On the host of the selected runtime database, run:

   ```
   dbbackup -y -t -n -x -c connection-string  target-directory
   ```

   where:

- **-y –** creates the backup directory or replaces a previous backup file in the directory without confirmation. If you want to be prompted, do not specify **-y**.
- **-t –** create a backup that can be used as an incremental backup since the transaction log can be applied to the most recently backed up copy of the database files.
- **-n –** changes the naming convention of the backup transaction log file to `*yymmddxx*.log`. *xx* are sequential letters ranging from **AA** to **ZZ**. This indexed file naming convention allows you to keep backups of multiple versions of the transaction log file in the same backup directory.
- **-x –** backs up the existing transaction log, deletes the original log, and then starts a new transaction log. Do not use this option if you are using database mirroring.

For example, this command creates the incremental transaction log backup file in the folder `B:\default`, in which the transaction log of `default.db` (cache database) is stored:

```
dbbackup -y -t -n -x -c
"dbn=default;uid=dba;pwd=sql;links=tcpip
(DoBroadcast=NO;VerifyServerName=NO;host=
localhost;port=5200)" B:\default
```

2. Ensure the incremental `*yymmddxx*.log` backup is created in the target directory selected.
3. Repeat steps 1 and 2 on each remaining database in the data tier node.

Using dbtran to Validate the Backup

Validation allows you to check the integrity of incremental backups and to ensure that you can safely apply these increments to the offline backup. This is a vital step and cannot be bypassed, without risking the viability and integrity of your backup data.

Use **dbtran** to translate the incremental transaction log generated to SQL commands. This utility fails if any data in the log is invalid or corrupted. If the utility succeeds, you can safely apply the log to the full backup on the recovery server.

1. Validate the transaction log backup by converting it to SQL command file:

```
dbtran yymmddxx.log yymmddxx.sql
```

2. Delete the resulting `*yymmddxx*.sql` file as it is not required for any other purpose.

Applying Increments to Full Backups on the Recovery Server

Update the full backup with contents of the transaction log.

Prerequisites

Use **dbtran** to ensure that the transaction log backup has been validated..

Task

These steps assume that the backed up files are available on the same drive as the one used by the production nodes used in the data tier node.

You can find all utilities in the `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere12\BIN32` folder.

1. Apply the incremental transaction log to the full backup by running:

   ```
   dbeng12 -o recovery.log db_name.db -a file\yymmddxx.log
   ```

   This command updates the full backup file (`db_name.db`) to the current state of the transaction log backup. This is the new baseline full backup from which you can recover failed environments. To update the backup again, reapply the next increment.

   Note: Database files (for example, `default.db`, or `clusterdb.db`) are stored in a parent folder, whereas transaction logs are stored in a subfolder of the same name as the database (for example, for `default.db` the transaction log is output to `..\default\120616.log`.

2. Replace the `db_name.db` in the data directory of the restore server and restart Unwired Platform data services.
3. Verify the full backup by running:

   ```
   dbvalid -c connection-string
   ```

See also

- *Performing Periodic Incremental Transaction Log Backups Online* on page 80

Maintaining Backups

Carefully maintain your backup artifacts, so you can use them to recover your data in the event of a hardware failure or other system problem that causes data loss.

See also

- *Planning Backup and Recovery* on page 74
- *Backing Up Runtime Data with a Recovery Server* on page 77
- *Diagnosing and Recovering from Failure by Scenario* on page 83
- *Performing Postrecovery Activities* on page 85

Monitoring the Incremental Backup State

Monitoring the completion of your Sybase Unwired Platform database backups is critical to success of your recovery plan.

Monitoring the incremental backup state is not automated. You need to independently perform these investigations unless you have a third-party monitoring system you can use for this purpose.

1. (Only required if using log mirrors) Always check the the log mirror, to ensure there are no lags and that they are exact replicas of the source transaction log at a given moment.

2. If you are using a third-party tool for performing the incremental backup, check that the scheduled jobs are running and that the transaction logs are successfully applied to the backup database.
3. Perform routine verification of the backup by running:

   ```
   dbvalid -c connection-string
   ```

Performing Ad Hoc Backup and Recovery Tests

Perform regular testing of the recovery plan for your Unwired Platform infrastructure in a lab environment that mimics your production environment. These tests allow you to discover the length of time needed to recover your system and any potential issues with the process or strategy you have created.

Disaster recovery tests may be required to comply with corporate business continuity policies. Sybase does not require testing as part of runtime maintenance, but recommends it as part of a larger continuity process.
Test the recovery process either by:

- Validating that the backup and recovery process are working as expected, and that the documentation describing the process is valid and readily available in case of a disaster.
- (Recommended) In a testing environment, perform an actual failover to the backup database, then failback to the primary database after the test is completed. Do not perform this test in a production environment.

Diagnosing and Recovering from Failure by Scenario

Not all issues or failures require recovery. Review the list of documented scenarios to determine whether to initiate the recovery process.

See also

- *Planning Backup and Recovery* on page 74
- *Backing Up Runtime Data with a Recovery Server* on page 77
- *Maintaining Backups* on page 82
- *Performing Postrecovery Activities* on page 85

Database Failure

Problem: Only the database has failed; the database server and the media are still functioning.

Symptoms: You can diagnose a failed database by finding some `database access` errors in Unwired Server logs. Or testing the database with **dbvalid** results in a failure.

Recovery required

1. Stop all Sybase Unwired Platform services on all Unwired Server and data tier nodes in the cluster.
2. Apply the transaction log to the backup database.

3. Replace the primary database with the backup database.

Hardware Failure: Mirrors Available

Problem: The storage media or host has failed; no database file nor log file is available. Transaction log mirroring was enabled.

Symptoms:

Recovery required:

1. Stopping all Sybase Unwired Platform services.
2. Applying the log mirror to the backup database.
3. Replacing the primary database with the backup database.

Hardware Failure: No Mirrors Available

Problem: The storage media or host has failed; no database file nor log file available. Transaction log mirroring is not enabled.

Symptoms:

Recovery required:

1. Stop all Sybase Unwired Platform services.
2. Replace the primary database with the backup database.

You can expect some data loss for transactions that occurred since the last incremental backup was created.

Backup Database Is Lost

Problem: The backup database is lost. While this is not a production failure it does pose a risk.

Symptoms: No back up is available.

Recovery required:

1. Stop all Sybase Unwired Platform services.
2. Perform a new full offline database backup to create a new backup file.

Database Corruption

Problem: You suspect the database is corrupt, and want perform a verification to assess this concern.

Verifcation required:

To verify corruption, run:

```
dbvalid -c connection-string
```

If **dbvalid** returns anything other than a `No errors reported` message, restore the full database using the backup on the recovery server.

Recovering from Server Node Software Corruption or Host Failure

Problem: The installed software on a node becomes corrupted or the host computer fails..

Recovery required:

1. Ensure Unwired Server services on the data tier node are running. In versions of Unwired Platform 2.1 ESD #2 and earlier, the runtime services must also be running on a primary cluster node.
2. If you are recovering a software component with the installer:
   a) Uninstall the affected component.
   b) Reinstall the affected component.
   c) If the component is a data tier component, copy the database file from the recovery server.
3. If you are recovering a software component with a clone or snapshot image, revert to the most recent backup.

   Note: If reverting to a backup fails, or if Unwired Server services do not start after being restored, or if a backup is unavailable, you must reinstall the required components on the failed node.

See also

- *Applying Increments to Full Backups on the Recovery Server* on page 81

Performing Postrecovery Activities

Once you have performed the appropriate recovery process as determined by the failure scenario, there may be some postrecovery activities that must be performed, either by the device user or by the administrator.

See also

- *Planning Backup and Recovery* on page 74
- *Backing Up Runtime Data with a Recovery Server* on page 77
- *Maintaining Backups* on page 82
- *Diagnosing and Recovering from Failure by Scenario* on page 83

Synchronizing Data from Device Applications

If the database fails after device users have synchronized since the last successful backup (and both the primary database and mirror logs are lost), synchronization may fail. Incomplete transactions must be resubmitted once the client database is reset.

Unwired Server asks the client to send a new upload that starts at the last known synchronization point. However, because of differing synchronization timestamps, cache and client databases disagree on when the last synchronization took place because progress offsets do not match. This mismatch requires a data purge, and often results in some data loss. Users must re-create data changes need and resubmit them.

1. **1.** Validate this issue by checking these logs after the failed synchronization attempt:
   - In the device client log, look for a `Synchronization server failed to commit the upload` error.
   - In the Unwired Server, look for messages relating to progress offset disagreements.

     ```
     Config16VM2-server.log:2012-03-12 18:17:12.287 WARN Mobilink
     Thread-170 [com.
     sybase.ml.sup.Logger] [10012] The consolidated and remote
     databases disagree on when
     the last synchronization took place, the progress offsets are 9
     in the consolidated
     database and 12 in the remote database. The remote is being
     asked to send a new
     upload that starts at the last known synchronization point.
     ```
2. **2.** Resolve this issue by asking the device user to:
   a) From the application, initiate a database reset.

   Note: Developers can include the reset functionality by calling `DB.DeleteDatabase` in the application code.

   b) Restart the application.
   c) Perform an initial sync.
   d) Re-create any pending changes and resynchronize them to Unwired Server.

See also

- *Resubmitting Lost Transactions* on page 86
- *Reregistering Applications* on page 87

Resubmitting Lost Transactions

Transactions that are not committed to the EIS before a cache database failure must be resubmitted by the application user.

1. **1.** Validate that this is the issue by comparing the data state on the device with the data state of the EIS backend system data.

   Note: Other symptoms that are based on the actual application design may exist; work with the development team to evaluate those application specific symptoms that may arise.
2. **2.** Resolve the lost transaction issue by requesting that the user:
   a) Reregister the application connection.
   b) Resubmit all transactions from the application after it is reregistered.

See also

- *Synchronizing Data from Device Applications* on page 85

- *Reregistering Applications* on page 87

Reregistering Applications

If a device is registered between backup and failure events, Unwired Server has no registration information for that device after recovery.

1. Validate the issue by checking Unwired Server logs for this message: `Invalid or Expired User Name and/or Authentication Code`.

   This message shows that an unregistered device has tried to connect. No message is sent or received between this device and Unwired Server.
2. Resolve the issue by asking the device user to reregister the application connection.

See also

- *Synchronizing Data from Device Applications* on page 85
- *Resubmitting Lost Transactions* on page 86

Backup and Recovery of Security Artifacts

The Unwired Server runtime uses keys and certificates that need to be backed up. Back up the security artifacts, to prevent key mismatch errors with the runtime and device applications.

Backing Up and Recovering Messaging Keys

During startup, Unwired Server uses a messaging key that is created only at messaging service startup, then saved in the Windows registry. The registry entry is Base64 encoded, and the contents must be used as the value for the `serververificationkey=` key in messaging applications or Hybrid Apps.

Registry entries for the messaging keys are recorded in `Computer\HKEY_LOCAL_MACHINE\Software\Node\Sybase\SybaseMessagingServer\Server\identinfo`. If you are using an alternate method for creating complete backups of a host system, continue to use that method to backup and recover these keys.

1. To backup required messaging service registry entries, from a command prompt, run:

   ```
   regedit /s /e msgserver.reg "HKEY_LOCAL_MACHINE\SOFTWARE
   \Wow6432Node\Sybase\Sybase Messaging
         Server\Server\identinfo
   ```

   A file is exported to `C:\Windows\System32\msgserver.reg`. This file is human readable, and contains the RSA private key that is fundamental to secure messaging.
2. Save the exported registry to a secure location.

Note: If this key becomes compromised, the security of messaging communications can no longer be guaranteed.

3. Delete the `C:\Windows\System32\msgserver.reg` file.
4. To recover a system that has failed:
   a) Reinstall Unwired Platform.
   b) Stop Unwired Server.
   c) Copy the `msgserver.reg` file from its secure location to a temporary location on the server that is being restored (for example `C:\temp`).
   d) From a command prompt, run:

      ```
      regedit /s <filePath>\msgserver.reg
      ```

      For example, if you saved the file to `C:\temp` then the command is:

      ```
      regedit /s C:\temp\msgserver.reg
      ```

   e) Delete the `msgserver.reg` file from it's temporary location.
   f) Restart Unwired Server.

Backing Up and Recovering Other System Keys and Certificates

All other platform keys and certificates are stored in the keystore and truststore respectively. These stores should be backed up with the entire installation file system.

1. Regularly perform backups of the entire installation directory as part of your disk backup schedule.
2. To recover other system keys and certificates, locate the file system backup and retrieve the contents of the `< UnwiredPlatform_InstallDir >\Servers \UnwiredServer\ \Repository\Security` folder.

CHAPTER 15 Platform Monitoring and Diagnostics

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 the platform.

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 10. 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.

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 288

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
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.
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 11. 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.
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 12. 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.

Category	Description
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.

KPI	Description
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 13. Detail view information

Data type	Description
Application ID	The ID number associated with an application.
Package	The package name.

Data type	Description
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 14. 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 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

Category	Description
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 synchro-nization; for example, subscribe, recover, suspend, and so on
Total Errors	The number of errors that occurred for the package during syn-chronization
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.
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.

KPI	Description
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.

Category	Description
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.
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.

KPI	Description
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 synchroniza-tions.
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 Hybrid App 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 Hybrid App 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, or data change notification)
Last Invalidate Time	The last time the cache was invalidated
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 see *Developer Guide: Unwired Server Runtime > 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**.

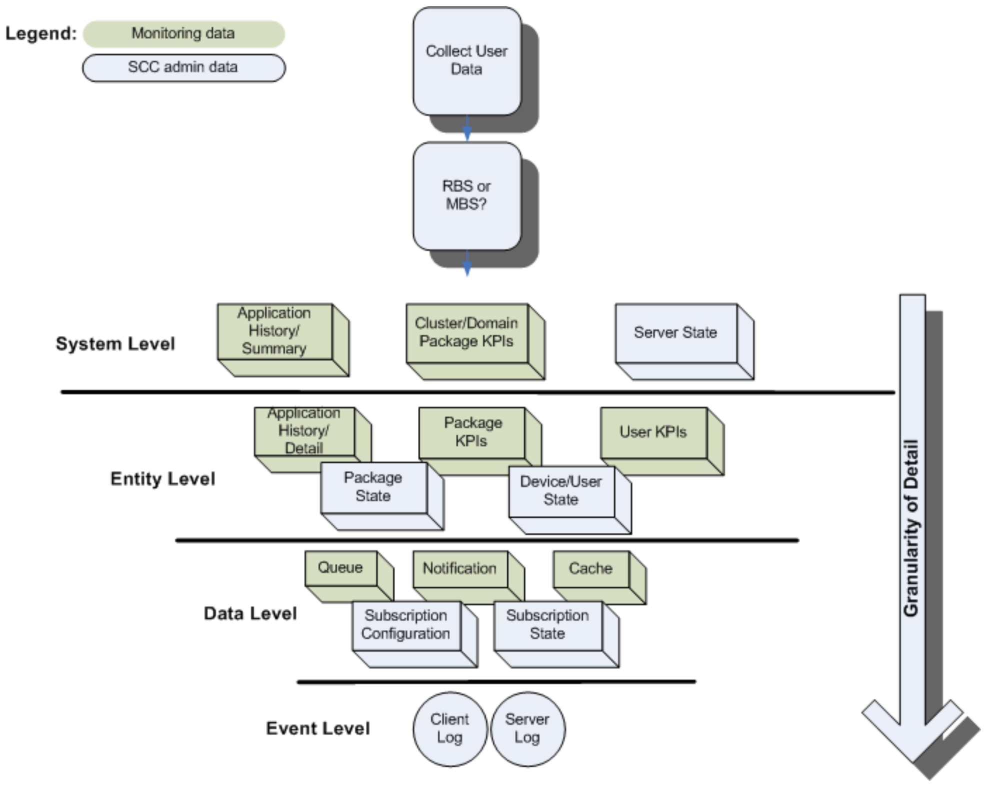
7. In the file browser dialog, select a save location and enter a unique file name.
8. Click **OK**.
   All monitoring data retrieved by the search is saved to the file you specify in step 7.

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 118
- *Access Denied Analysis* on page 122
- *Data Update Failure Analysis* on page 123

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 perform-ance issues	Package historical data to see if high volumes or frequent operations are causing the issue or problem	*Checking Package/User Histor-ies*
	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 ad-ministration data to see the respon-siveness of the server	*Checking Server Responsive-ness in Sybase Control Center*
Unwired Server errors or failures	System logs to see if there are mes-sages indicating whether something has failed	*Looking for Errors and Failures in SCC*

See also

- *Collecting Data* on page 118
- *Checking Package/User Histories* on page 119
- *Checking Overall Package Statistics* on page 120
- *Checking User KPIs for Other Data Usage* on page 120
- *Checking Server Responsiveness in Sybase Control Center* on page 121
- *Looking for Errors and Failures in SCC* on page 122

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 administrator needs to first assess the synchronization performance by checking data as documented in *Device Application Performance or Issue 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 change detection interval is configured appropriately based on the cache interval or schedule repeat. This value determines how frequently change detection occurs. If you see a value, of 0, then this action is disabled. Enable this action by setting an appropriate value.
   - 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 15. Log file locations

Log type	Location
Unwired Server	Aggregated Unwired Server logs: `SUP_HOME\Servers\UnwiredServer\logs` Unwired Server log: `SUP_HOME\Servers\UnwiredServer\logs\host_server.log` Database error log: `SUP_HOME\Servers\UnwiredServer\logs\errorlog.txt` Message Server (MobiLink) error log: `SUP_HOME\Servers\UnwiredServer\logs\mlsrv_err.log` Bootstrap log: `SUP_HOME\Servers\UnwiredServer\logs\bootstrap*.log` Messaging service log details: `SUP_HOME\Servers\UnwiredServer\logs\Module` Device client tracing logs: `SUP_HOME\Servers\UnwiredServer\logs\ClientTrace` Hybrid App tracing logs: `SUP_HOME\Servers\UnwiredServer\logs\WorkflowClient`
Sybase Control Center for Sybase Unwired Platform	Sybase Control Center agent log: `SCC_HOME\log\agent.log` Gateway log: `SCC_HOME\log\gateway.log` Repository log: `SCC_HOME\log\repository.log` Request logs: `SCC_HOME\services\EmbeddedWebContainer\log\request-<yyyy_mm_dd>.log` Database server log: `SCC_HOME\services\Repository\scc_repository.slg`

Log type	Location
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: *SUP_HOME*`\Servers\UnwiredServer\logs\<`*nodeName*`>.RSOE<`*n*`>.log` For example: `RBShost1.RSOE4.log`.

Installer	`SUP_HOME`
	`SUP_HOME\InstallLogs`
	`SUP_HOME\InstallLogs\silentInstall`

ESD Installation Log	`SUP_HOME\sup`*XX*`ebf`*X*`logs`
Domain	In the domainlog database; viewable in Sybase Control Center.
Cache database logs	By default, cache database errors are logged in the `errorlog.txt` file, located in *SUP_HOME*`\Servers\UnwiredServer\logs`. For a data tier installation, there are three database server logs, by default located in *SUP_HOME*`\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 Sybase 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 Unwired Server Log Settings* in *Sybase Control Center for Sybase Unwired Platform*.

Log messages that typically get recorded include:

Level	Description
Debug	Detailed information useful for debugging purposes.
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 high level, 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.
- Messaging Server logs – retrieve trace data for all or specific messages. Export data for archive or further analysis.

Unwired Server Runtime Logging

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

You can change default log levels for different components as required from Sybase Control Center.

You can view logs from:

- Sybase Control Center – click **Servers** > ***primaryServer*** > **Log** in the left 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.
- 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 server's log file.

Configuring Unwired Server Log Settings

Unwired Server logs collect data on Unwired Server health and performance by component. 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.

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 **Configuration**.
2. In the right administration pane, click the **Log Setting** tab and select **Unwired Server**..
3. 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.
4. Set the MMS 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.
5. Set the HTTP log settings.
   a) Select **Enable HTTP request log** to generate an HTTP request log in the logs subdirectory. The generated log file name is `server-name-http.log`.

Note: HTTP logging is off by default. Enabling HTTP logging can cause a performance impact and possible logging of sensitive data.

b) Set the **Maximum file size** of the log file.

c) If you want to back up the log file when it reaches the limit, select **Perform rotation**. The backup file is saved as `*server-name*-http.log.*backup-number*`.

d) If you want to continue to use the current log file when the server restarts, select **Reuse**.If you do not select this option, when the server restarts the current log file is copied to the `.\old` subdirectory and a new log file is created.

e) If you want to archive the log file select **Archive**, then specify the **Archive file name**. If you want to compress the archive log file select **Compress**.

6. For each component, choose a log level:

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

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

Log level	Messages logged
Console	Messages that appear in the administration console only (when Unwired Server is running in non-service mode)
Off	Do not record any messages

7. 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 135
- *Configuring RSOE Logging* on page 134
- *Enabling Custom Log4j Logging* on page 137

Messaging Server Runtime Logging

(Does not apply to Scale Out server nodes) Messaging Server logs collect data that enables you to trace message handling from the cluster database to the device user, based on various trace settings.

Configuring Messaging Server Log Settings

Messaging Server logs create trace configurations for messaging modules, and retrieve trace data for all or specific messages. Configure trace configuration properties for modules to specify the amount of detail that is written to the log. You can configure trace settings for the primary server cluster in Sybase Control Center for each module. The settings are available to cluster servers through the shared data folder.

Note: The default settings may only need to change in case of technical support situations where, for diagnostic reasons, a request is made to configure the specific module(s) settings, and provide the request log. In all other cases, the administrator or developer should not need to change the settings.

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 **Configuration**.
2. In the right administration pane, click the **Log Setting** tab and select **Messaging Server**.
3. Select Default, or one or more of the messaging service modules. Click **Show All** to show all modules.

Module	Description
Default	Represents the default configuration. The default values are used if optional fields are left blank in a module trace configuration. Required.
Device Management	Miscellaneous functions related to device registration, event notification, and device administration. Enable tracing for problems in these areas.
JMSBridge	This module handles communications from the Unwired Server to the messaging server. Enable tracing to view the detailed messaging exchange.
MO	This module handles the delivery of messages between the client and server, including synchronous function calls from client to server. Enable tracing for MO errors and message delivery issues.
SUPBridge	This module handles communications from the messaging server to the Unwired Server. Enable tracing to view the detailed messaging exchange.

Module	Description
TM	This module handles the wire protocol, including encryption, compression, and authentication, between the messaging server and clients. All communication between the client and the messaging server passes through TM. Enable tracing for authentication issues, TM errors, and general connectivity issues.
WorkflowClient	The WorkflowClient module.

4. Click **Properties**.
   a) Enter trace configuration properties. If you selected multiple modules, a string of asterisks is used to indicate settings differ for the selected modules. You can select the option to view or change the property value for any module.

Property	Description
Module	Display only. Default, module name, or list of module names selected.
Description	(Optional) Custom description of the server module.
Level	Trace level for the module - DISABLED, ERROR, WARN, INFO, DEBUG, DEFAULT. If the default trace level is specified for the module, the module uses the trace level defined for Default. Required.
Max trace file size	(Optional) Maximum trace file size in MB. If the trace file size grows larger than the specified value, the trace file data is backed up automatically.
User name	(Optional) Only data for the specified user name is traced.
Application Connection ID	(Optional) Only data for the specified Application ID is traced.

   b) Click **OK**.
   Log files for each module are stored in folders of the same name located in: `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs`.

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 `SUP_HOME\Servers\UnwiredServer\logs\<server.name>-http.log`.

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

1. Open `SUP_HOME\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 `SUP_HOME\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 135
- *Configuring Unwired Server Log Settings* on page 129
- *Enabling Custom Log4j Logging* on page 137

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 administration 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 `SCC_HOME\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 `SCC_HOME\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: 25MB) and maximum backup index (default: 20 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 Sybase Control Center.

   The `executionTime.log` file now appears in the `SCC_HOME\log` folder.

Alternately, you can use the Adobe Flex log to track performance in Sybase Control Center:

1. Modify the `SCC_HOME\plugins\com.sybase.supadminplugin\agent-plugin.xml` file as indicated in step 2, above.
2. Restart Sybase Control Center.
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 129
- *Configuring RSOE Logging* on page 134
- *Enabling Custom Log4j Logging* on page 137

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: `SUP_HOME\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 135
- *Configuring Unwired Server Log Settings* on page 129
- *Configuring RSOE Logging* on page 134

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 `*SUP_HOME*\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 `*SUP_HOME*\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.

- Sybase Messaging Server
- SQLANY 12.0 (32 bit) or SQLANY64 12.0 (64-bit)
- SybaseUnwired Server
- Sybase Control Center *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 Domain Logging* in *Sybase Control Center for Sybase Unwired Platform*.

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 Hybrid App 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.
Push	Provides a few of push notification activities.
Proxy	Provides a view of Online Data Proxy connection-related activities.
Server	Provides a view of server-related activities.
Dispatcher	Provides a view of dispatcher activities. Within this subsystem, categories include Replication, Messaging, and Service.
Application	Provides a view of application activities. Within this subsystem, categories include Registration and Setting.

.

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 Logs* in *Sybase Control Center for Sybase Unwired Platform* 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 30

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.

If you do not configure the auto-purge schedule, you can purge data manually with the **Purge** button. If you are manually purging logs with hundreds of thousands of entries, note that Unwired Server removes these entries asynchronously to avoid negatively impacting runtime performance. For smaller logs, the purge action tends to be more instantaneous. To avoid large logs, use the auto purge schedule.

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. Optional. Change the maximum length of the payload data logged in the payload column(s) of each sub-system. Large payload content is truncated to the length specified as that value. The default max size is 12K (in bytes) which is configured in the 'default' domain and applicable for all domains. Increasing the domain payload size should be tested to identify proper configuration for the server's JVM memory settings.

10. 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.

Using End to End Tracing to Troubleshoot and Diagnose Device Problems

Follow this sequence to record device-initiated end to end trace information.

Prerequisites

1. The client must be configured to use end to end tracing by configuring the client to use the supportability-related tracing APIs, and the trace level must be set to a value other than NONE. For example, Unwired Server retrieves the appropriate properties set in the SAP Passport that are transported as an HTTP header in the request. For more information, see the *Developer Guide for your platform*.
2. For REST API applications, the business transactions configured for tracing must be uploaded using the Business Transaction XML (BTX). For more information, see the *Developer Guide for REST API Applications*.

Task

1. The device application user initiates tracing on the device.
2. The System Administrator creates a logging profile for a given application connection ID based on the users information.

   Enable end to end tracing by selecting the application connection ids of interest in the Domain Logging Profile's Application connections screen for new or existing domain logging profiles. The System Administrator can either:
   - Create a new domain logging profile from Sybase Control Center
     a. Select **Domains > Log > Settings > New screen**. In the profile definition dialog.
     b. Select **Application Connections**, select **Application Connection ID** as your search criteria, select the application connections you intend to trace and click **OK**.
     c. Select **Enable after creation** and click **OK**.
   - Modify an existing application connection profile by selecting and editing an existing domain logging profile to include the application connections you want to trace in the "Application connections" filter of the profile. Verify that the domain logging profile is enabled.
3. The user turns off tracing on the device.
4. The System Administrator filters the server log by root context ID and transaction ID, as provided by the application.

   The transaction id and root context id fields are used to trace a given request. The field values should be saved in all modules where trace data is required. Each request made to the server has a different transaction id and all requests within a given session have the same root context id. Each such session has different root context ids. For example, consider the following snippet of client code:

```
E2ETraceTestDB.Synchronize();
int id = 121;
```

```
Customer customer = new Customer();
customer.Fname = "Ritchie";
customer.Lname = "Dennis";
customer.Id = id;
customer.City = "CA";
customer.Save();
customer.SubmitPending();
E2ETraceTestDB.Synchronize();
```

For the above code, the log messages in a typical domain log table can be represented as given below. Here the logs resulting from each `synchronize()` server call includes a unique `TransactionID` but both calls include the same `RootContextID`:

Table 16. Transaction and Root Context ID Logging

LogID	RootContextID	TransactionID	LogMessage
1	4635000000311EE09EC5037310AE97A2	4635000000311EE09EC5037310AE77A2	Msg1 for sync1
2	4635000000311EE09EC5037310AE97A2	4635000000311EE09EC5037310AE77A2	msg2 for sync1
3	4635000000311EE09EC5037310AE97A2	4635000000311EE09EC5037310AE77A2	msg3 for sync1
n+1	4635000000311EE09EC5037310AE97A2	4635000000311EE09FG5037310BD99A2	msg1 for sync2
n+2	4635000000311EE09EC5037310AE97A2	4635000000311EE09FG5037310BD99A2	msg2 for sync2

5. View the server log. The trace level set in the passport by the client devices determines the amount of end to end tracing captured in these logs:

- Domain logs
- Payload logs
- Performance logs
- Sybase Unwired Platform log level

End to end trace levels are determined by the application's trace level setting:

Table 17. Trace Levels

Trace Level	Enable Domain Log	Enable Payload Log	Enable Performance Log	Override Sybase Unwired Platform Log Level
NONE	No	No	No	No
LOW	Yes	No	No	INFO
MEDIUM	Yes	No	Yes	INFO
HIGH	Yes	Yes	No	DEBUG

When the trace level is NONE, the system behaves as if SAP Passport is not set and end to end tracing is disabled. The overridden configuration is in effect only for the current thread (context), so the configuration remains unaffected for other application connections for which trace is not enabled. When end to end tracing is enabled, the log level set against all buckets (modules) in Sybase Unwired Platform is overridden to the one given above as determined by the trace level.

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 `SCC_HOME\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 `SCC_HOME\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 `SCC_HOME\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 `SCC_HOME\plugins\ com.sybase.supsnmpplugin_X.X.X\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 `SCC_HOME\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.

Workload Analysis with Wily Introscope Agent

Introscope is a third-party tool that can be integrated into your system landscape to quickly isolate and resolve performance issues wherever they arise in each stage of the application lifecycle. Administrators and support personnel can use performance data points for several key performance indicators (KPIs) to assess overall performance of the system.

Introscope is not installed by Sybase Unwired Platform; Introscope must be purchased and installed separately, before you can use this method of monitoring. The Unwired Platform only installs two agents with Unwired Server:

- Java agent (for use with scale-out or application server nodes)
- .NET agent (for use with application server nodes only)

Configure then enable the appropriate agent to ensure the proper data is tracked and uploaded to the Introscope tool.

Configuring and Enabling Introscope Agents for an Unwired Server

Use the `configure-introscope-agents.bat` utility to configure then enable an agent for the type or server node it supports. When the agent is no longer required, disable it.

Prerequisites

Do not configure or enable an agent unless Introscope is already installed in the production environment.

Task

The `configure-introscope-agents.bat` utility is installed to *`SUP_HOME`*`\Servers\UnwiredServer\bin`. Help is available with the `-h` option.

1. At a command prompt, run the utility to configure the Introscope agent to communicate over the required port with the targeted Unwired Server:

   ```
   configure-introscope-agents.bat -host:serverNodeName -
   port:6001
   ```

2. Enable the configured agent to allow KPIs from Unwired Server to be sent to Introscope.

   ```
   configure-introscope-agents.bat -install
   ```

3. To stop communicating KPIs, disable the agent:

   ```
   configure-introscope-agents.bat -remove
   ```

Key Performance Indicators

The Introscope Agent collects KPIs from Unwired Server. These KPIs are used to assess the health of Unwired Server.

KPIs vary for the agent type (Java or .NET) used.

Table 18. Java Introscope Agent Key Performance Indicators for Replication Applications

KPI Grouping	Description	Example
Authentication	Enumerates user authentication performed against a single login module. The KPI is a direct reflection of the execution time for the authentication attempt against the module type used. If a security configuration contains multiple login modules, a single user authentication result may occur in more than a one event.	`Sybase Unwired Platform|Security|{SecurityConfigurationName}|Authentication|{LoginModuleType}`
Registration	Enumerates all automatic application connection registration events.	`Sybase Unwired Platform|Applications|Registration`
Replication Events		
	The average time uploading data from the client application to the cache. This KPI does not include network time.	`Sybase Unwired Platform|{domain}|Cache|{package}|Synchronization|UploadData`
	The time processing entity operations in the package until the download starts.	`Sybase Unwired Platform|{domain}|Cache|{package}|Synchronization|PrepareForDownload`
	The amount of time fetching rows which need to be downloaded from the cache. This does not include network time.	`Sybase Unwired Platform|{domain}|Cache|{package}|Synchronization|DownloadData`

KPI Grouping	Description	Example
	The average time processing messages for asynchronous cache operations and putting the message on the messaging sub system queue for transport. This does not include network time and the time spent by messaging subsystem.	`Sybase Unwired Platform|Cache|ProcessReplayBatchMessage`
	The event of generating internal data change event notifications.	Sybase Unwired Platform|{domain}|Cache|{package}|ScheduledNotify
	The average time generating internal data change event notifications.	`Sybase Unwired Platform|Cache|{classname}|{method}|Cache|{classname}|{method}`
Push/Server-initiated Synchronization (SIS)		
	Determines which application connections are affected by push changes since the last run of this task.	`Sybase Unwired Platform|{domain}|Cache|{package}|ScheduledNotify`
	The average time creating SIS notifications.	`Sybase Unwired Platform|{domain}|Cache|{package}|ScheduledNotify|CreateNotifications`
	The average time creating a single SIS notification.	`Sybase Unwired Platform|{domain}|Cache|{package}|ScheduledNotify|CreateOneNotification`
	The average time taken for Unwired Server to send notifications to clients over the messaging channel.	`Sybase Unwired Platform|{domain}|Cache|{package}|ScheduledNotify|InternalTransport`

KPI Grouping	Description	Example
	The average time taken for Unwired Server to send notifications to clients over the HTTP channel	`Sybase Unwired Platform|{domain}|Cache|{package}|ScheduledNotify|HttpTransport`
Messaging Synchronization		
	The average time processing all incoming asynchronous messages from messaging applications.	`Sybase Unwired Platform|Cache|Messaging|ProcessIncomingMessage`
	The average time processing messages related to bulk subscribe.	`Sybase Unwired Platform|Cache|Messaging|ProcessBulkSubscribeMessage`
	The average time determining which application connections are affected by changes since the last run of this task.	`Sybase Unwired Platform|{domain}|Cache|Messaging|{package}|ScheduledPush`
	The average time spent in background workerthread processing "persistent" events that will lead to import message generation for cache messaging based applications.	`Sybase Unwired Platform|Cache|Messaging|GenerateOutgoingMessages`
	The average time producing the actual push job (GenerateOutgoingMessages) to run in the context of a particular application connection for which data is to be delivered.	`Sybase Unwired Platform|{domain}|Cache|Messaging|{package}|ScheduledPush|CreatePushJobs`

KPI Grouping	Description	Example
Data Services (EIS read or write)		
	The average execution time for a single EIS operation of an entity.	`Sybase Unwired Platform|{domain}|Cache|{package}|{mbo}|{operation}`
	The time spent modifying the cache with the supplied rows of an entity.	`Sybase Unwired Platform|{domain}|Cache|{package}|{mbo}|ModifyCache`
	Determines the degree of change. The merge processor only updates the cache incrementally when the EIS partition is empty, or the cache partition is empty and there are no foreign keys or the cache is unpartitioned. Otherwise, the change commands are collected and executed in series as a batch when applyDeltas() is invoked.	`Sybase Unwired Platform|{domain}|Cache|{package}|{mbo}|DetermineDelta`
	For the specified partition, this method compares the contents of the cache with the contents of the partition in the EIS; any changes are detected by computeDeltas() and translated into change commands.	`Sybase Unwired Platform|{domain}|Cache|{package}|{mbo}|ApplyDelta`
Data maintenance		

KPI Grouping	Description	Example
	The average time spent removing logically deleted rows in the cache that are older than the oldest synchronization time on record in the system. This cleanup task also removes unused or stale partitions.	`Sybase Unwired Platform|{domain}|Cache|{package}|Data Maintenance|PurgeEntities`
	The average time spent removing client log records that have already been synchronized to the device, or are no longer associated with active users.	`Sybase Unwired Platform|{domain}|Cache|{package}|Data Maintenance|PurgeClientLog`
	The average time spent removing historical data on MBO data refresh and operation replay failures, which result from system or application failures.	`Sybase Unwired Platform|{domain}|Cache|{package}|Data Maintenance|PurgeErrorLog`
	The average time spent in removing subscriptions that are not active for the 'number of inactive days' in the schedule task configuration.	`Sybase Unwired Platform|{domain}|Cache|{package}|Data Maintenance|PurgeSubscriptions`
DCN incoming events from EIS for cache manipulation or HWA trigger		
	The average time spent in parsing and processing the DCN request for Hybrid Apps on Unwired Server.	`Sybase Unwired Platform|{domain}|Hybrid App Notification|ProcessRequest`

KPI Grouping	Description	Example
	The average time spent in parsing and processing the DCN request on Unwired Server.	`Sybase Unwired Platform|{domain}|Cache|{package}|Data Change Notification|ProcessRequest`
Messaging		
	The average time processing incoming messages over IIOP.	`Sybase Unwired Platform|MessageChannel|Transport`
	The average time executing commands by a registered Handler.	`Sybase Unwired Platform|MessageChannel|Dispatcher`
	The event of a single MBO request invocation from a Hybrid App.	`Sybase Unwired Platform|MessageChannel|MboRequestHandler`
	The event of requesting a specific customization resource bundle from Unwired Server.	`Sybase Unwired Platform|MessageChannel|CustomizationResourceHandler`
	The event of uploading a business transaction in XML from the device to Unwired Server, including the actual time to transfer the file to the Solution Manager.	`Sybase Unwired Platform|MessageChannel|BtxUploadHandler`
	The average time inserting the device logs into the domain log database.	`Sybase Unwired Platform|MessageChannel|ClientLogsHandler`
	The event of logging a trace record in domain log database by messaging sub system.	`Sybase Unwired Platform|MessageChannel|E2eTraceHandler`
Online Data Proxy		

KPI Grouping	Description	Example
	The time spent handling a single client call from an application in the messaging channel.	`Sybase Unwired Platform|Proxy|MessageChannel`

	The time interacting with the backend in the messaging channel.	`Sybase Unwired Platform|Proxy|MessageChannel|EndpointCall`
		`Sybase Unwired Platform|Proxy|MessageChannel|EndpointCall|GatewayUrl`

	The average time processing the response from the backend in the messaging channel.	`Sybase Unwired Platform|Proxy|MessageChannel|PrepareResponse`
	The time spent handling a single client call from an application in the HTTP channel.	`Sybase Unwired Platform|Proxy|HttpChannel`

	Invokes the backend in the HTTP channel.	`Sybase Unwired Platform|Proxy|HttpChannel|EndpointCall`
		`Sybase Unwired Platform|Proxy|HttpChannel|EndpointCall|GatewayUrl`

	The average time processing the response from the backend in the HTTP channel.	`Sybase Unwired Platform|Proxy|HttpChannel|PrepareResponse`
Native Push Notifications		
	The event of processing the push notification message delivery to native push subsystems.	`Sybase Unwired Platform|NativePush|POST`

KPI Grouping	Description	Example

	The average time sending a notification to native notification servers. This does not include time spent by native notification servers to send the message to actual device.	`Sybase Unwired Platform|NativePush|ProcessNotification|APNS`
		`Sybase Unwired Platform|NativePush|ProcessNotification|BES`
		`Sybase Unwired Platform|NativePush|ProcessNotification|GCM`

Table 19. .NET Introscope Agent Key Performance Indicators for Messaging or Hybrid Apps

KPI Grouping	Description	Example

Incoming messages from applications to Unwired Server	The average time spent in sending the request from the messaging servers to Unwired Server.	`Sybase Unwired Platform|Cache|Messaging|MessageFromClient`
		`Sybase Unwired Platform|Cache|Messaging|MessageFromClient:Errors Per Interval`
		`Sybase Unwired Platform|Cache|Messaging|ClientErrorCallback|{1}:Errors Per Interval`

Message channel	Generic method called by device -- performs web request (HTTP) of MMS for a synchronized method call over messaging channel for a particular Handler-Id call with respective domain and security configuration	`Sybase Unwired Platform|MessageChannel|Transport|{handler-id}-{domain}-{security configuration}`
Hybrid Apps		
	The event of executing the synchronous request from the Hybrid App.	`Sybase Unwired Platform|Hybrid Apps|ExecuteRequest`

KPI Grouping	Description	Example
	The event of executing the asynchronous request from the Hybrid App.	`Sybase Unwired Platform|Hybrid Apps|SubmitResponse`
	The event of executing the object query (if any) and putting the message response into the messaging queue.	`Sybase Unwired Platform|Hybrid Apps|ProcessEvent`
Outbound messages from Unwired Server to application		
	The event of submitting the message from Unwired Server to the device outbound queue.	`Sybase Unwired Platform|Cache|Messaging|MessageFromServer|Process`
	The average time taken for the server to acknowledge the message after it was put into the outbound queue.	`Sybase Unwired Platform|Cache|Messaging|MessageFromServer|Acknowledge`

CHAPTER 16 SAP Interoperability

In an interconnected SAP-driven computing ecosystem, enabling interoperability between products is an important operations management task that enables Unwired Platform runtime data sharing with these SAP systems.

Different SAP systems have different methods of data exchange.

SAP SLD Server Overview

For SAP environments that use Solution Manager for runtime root-cause analysis, configure a destination System Landscape Directory (SLD) server. This configuration allows Unwired Platform to deliver runtime information to a common SAP SLD repository, keeping information about your SAP and Unwired Platform mobility infrastructure complete and current.

Table 20. SLD Administration Tasks

Task	Frequency	Perform in
Configure a new destination SLD server	One time	Sybase Control Center
Add, remove, or edit destination server connection properties	Infrequent, required as environment changes	Sybase Control Center
Export data	Infrequent	Sybase Control Center
Enable and disable the schedule	Routine	Sybase Control Center
Edit schedule properties	As required	Sybase Control Center
Upload XML payloads on demand	As required	Sybase Control Center

SLD and Unwired Platform Architecture

Unwired Platform uses an SLD destination and a scheduled task to generate required XML payloads and update them to the SLD server.

This cross-product communication and delivery is performed by various components of the Unwired Platform runtime:

- The primary Unwired Server is responsible for the cluster-level scheduling task. It also gathers local server information and routes data to the cluster database. When the schedule milestone is reached, the primary server generates and uploads the payload from information held in the cluster database.

- Any secondary Unwired Server automatically gathers local server information and routes data to the cluster database. The administrator need not configure this activity as it is controlled by the cluster-level task.
- The cluster database aggregates and holds information downloaded from all cluster server members.

See also

- *Sharing Cluster Data with the SLD* on page 162

Aggregating and Holding Cluster Data

Unwired Platform uses the cluster database to centralize the collection of all cluster data that becomes the payload to SLD servers.

Information is collected from each Unwired Server node that is a member of the identified cluster, and each node follows the schedule configured and enabled for the cluster. A single database transaction creates the database entry.

Once data is in the database, it is held until the next schedule milestone is reached, whereby a payload is generated and sent to the SLD destination server. At this point the data is purged from the database. When the next collection is requested by the schedule, a new set of cluster data is aggregated, held, delivered, and purged.

Note: During this process, if some nodes fail or are otherwise disabled (thereby preventing the collection task from being performed), the aggregated data currently held in the database generates the payload for those nodes.

Sharing Cluster Data with the SLD

Configure a cluster to connect to the System Landscape Directory (SLD) and generate and upload the payload that contains all details of the Unwired Platform system.

Prerequisites

Your system design must have taken SLD usage into account before you can configure its use in Unwired Platform. See *SLD Interoperability Requirements* in *Enterprise Mobility Landscape Design and Integration* .

Task

Choose the upload method you wish to use:

See also

- *SLD and Unwired Platform Architecture* on page 161

Uploading Payloads with Sybase Control Center

Use Sybase Control Center to register the SLD server, and then either upload generated payloads on-demand or with a configured (and enabled) schedule.

1. *Registering or Reregistering SLD Server Destinations*

   Registering an SLD destination identifies the connection properties needed to deliver the payload. You can register multiple destinations as required by your SAP environment. If your SLD server properties change, you must update properties as required and reregister the server with new values.
2. *Configuring and Enabling Scheduled Payload Generation and Uploads*

   Configure a schedule to automatically generate a new payload that uploads to an SLD server once cluster information is aggregated from all cluster members.
3. *Manually Uploading or Exporting Payloads On-Demand*

   Run an SLD payload generation task manually to generate and upload a payload on demand. Alternatively, export the payload to an XML file to archive SLD payload contents or to troubleshoot the cluster.

Registering or Reregistering SLD Server Destinations

Registering an SLD destination identifies the connection properties needed to deliver the payload. You can register multiple destinations as required by your SAP environment. If your SLD server properties change, you must update properties as required and reregister the server with new values.

For information about SLD, see *Configuring, Working with and Administering System Landscape Directory* on *http://www.sdn.sap.com/irj/sdn/nw-sld*.

1. In the navigation pane of Sybase Control Center, select the cluster name.
2. In the administration pane, click the **System Landscape Directory** tab.
3. Click **Servers**.
4. Choose one of the following:
   - If you are creating a new destination, click **New**.
   - If you are updating an existing destination, select the destination name in the table, and click **Properties**.
5. Configure the connection properties:

User Name	User name for the SLD server.
Password and Repeat Password	The user account password used to authenticate the user name entered. Password and Repeat Password must match for the password to be accepted.

Host	The host name or the IP address of the SLD server.
Port	The HTTP(S) port on which the SLD server is running. Enter a valid port number in the range of 0-65535.
Use secure	Select if you are using HTTPS protocol.

6. To validate the configuration, click **Ping**.
7. To accept validated configuration properties, click **OK**.

   This registers the SLD destination.

Configuring and Enabling Scheduled Payload Generation and Uploads

Configure a schedule to automatically generate a new payload that uploads to an SLD server once cluster information is aggregated from all cluster members.

1. In the navigation pane of Sybase Control Center, select the name of the cluster for which you want to schedule an SLD payload upload.
2. From the menu bar of the **System Landscape Directory** page, click **Schedule**.
3. To edit an existing schedule for a selected SLD server, click **Edit**.

   a) Configure the schedule:

   - **Schedule repeat** – select how often the schedule should run. Options are **monthly**, **weekly**, **daily**, **hourly**, and **custom**.
     - If you select **monthly** or **weekly**, specify:
       - **Start date** – select the date and time the automated upload should begin. Use the calendar picker and 24-hour time selector.
       - **End date** – select the date and time the automated upload should end.
     - If you select **daily** or **hourly**, specify:
       - **Start date** – select the date and time the automated upload should begin. Use the calendar picker and 24-hour time selector.
       - **End date** – select the date and time the automated upload should end.
       - **Days of the week** – select each day the automated upload schedule should run.
     - Select **custom**, to specify the interval granularity in seconds, minutes, or hours, as well as other date and time parameters.

   b) Click **OK**.
4. To enable the schedule, click **Enable**.

Manually Uploading or Exporting Payloads On-Demand

Run an SLD payload generation task manually to generate and upload a payload on demand. Alternatively, export the payload to an XML file to archive SLD payload contents or to troubleshoot the cluster.

1. In the navigation pane of Sybase Control Center, select the name of the cluster for which you want to immediately upload an SLD payload.
2. In the administration pane, click the **System Landscape Directory** tab.
3. From the menu bar of the **System Landscape Directory** page, click **Schedule**.
4. Click **Run Now**.
   The payload generation process begins.
5. Upon completion, review the contents of the payload and choose an action:
   - To export and save the contents to a file as XML, click **Save to File** and choose your file output name and location.
   - To upload the contents, select the target SLD servers and click **Finish**.

Uploading Payloads with Scripts

Use `runSLDReg.bat` to register the SLD server and generate payloads on-demand, before sending the payload contents with `sendPayLoad.bat`.

1. *Registering the SLD Server Destination with Scripts*

   Use the identified script to sett the connection properties used to send the generated payload.
2. *Generating and Uploading Payloads with Scripts*

   Use two scripts to generate and send the payload to System Landscape Directory (SLD).

Registering the SLD Server Destination with Scripts

Use the identified script to sett the connection properties used to send the generated payload.

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:

UserName	User name for the SLD server.
Password	The user account password used to authenticate the user name entered. Password and Repeat Password must match for the password to be accepted.
ServerHost	The host name or the IP address of the SLD server.

Port	The HTTP(S) port on which the SLD server is running. Enter a valid port number in the range of 0-65535.
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 and Uploading Payloads with Scripts

Use two scripts to generate and send 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:

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`.

SAP License Audit and Application Data Overview

For SAP applications like OData SDK Application, administrators can generate an XML file that contains usage audit data that is then sent to the SAP License Audit. The XML file, which is compatible with the License Audit infrastructure, includes counts of users using the applications currently deployed to Unwired Server.

Use Sybase Control Center to generate an audit measurement XML file for export to SAP License Audit.

Sharing Application Data with SAP License Audit

Generate an audit measurement file that includes usage data for Sybase Unwired Platform and usage data for SAP applications deployed to the server.

Generate an audit measurement file that includes license data related to application usage.

1. *Generating the SAP Audit Measurement File*

   Use Sybase Control Center to generate an audit measurement file.
2. *Uploading the SAP Audit Measurement File*

   Upload the audit measurement file to SAP License Audit by sending the file to SAP.

Generating the SAP Audit Measurement File

Use Sybase Control Center to generate an audit measurement file.

Using Sybase Control Center, generate a audit measurement file that can be sent to SAP for uploading to SAP License Audit.

1. In Sybase Control Center, select the Unwired Platform cluster and click the **General** tab.
2. Click **SAP Auditing Export**.
3. In the Export SAP Auditing Measurement window, enter the user name and click **Next**.
4. After Sybase Control Center generates the file, click **Finish**.
5. Select a save location for the file and click **Save**.

   Note: For information on uploading the audit measurement file to SAP License Audit, see supporting SAP documentation at *https://websmp108.sap-ag.de/licenseauditing*. Also see *Uploading the SAP Audit Measurement File*.

Uploading the SAP Audit Measurement File

Upload the audit measurement file to SAP License Audit by sending the file to SAP.

To upload the file to SAP License Audit, send the audit measurement file to SAP using the email address included in the measurement request from SAP. Included in this SAP-provided email is a link to the documentation for the SAP measurement process. See supporting SAP documentation at *https://websmp108.sap-ag.de/licenseauditing* .

SAP Applications Tracked with SAP License Audit

A list of applications that are tracked with SAP License Audit. The applications are registered or reregistered. All applications that have connections registered with Unwired Server can also have licenses counted by the SAP License Audit service.

Note: Applications created with Sybase Unwired Platform 2.1 or earlier are not counted.

ID	Unit
S001	SAP Mobile Platform User
S005	SAP Employee Lookup Mobile User
S010	SAP Leave Request Mobile User
S015	SAP Travel Receipt Capture Mobile User
S020	SAP Travel Expense Approval Mobile User
S025	SAP HR Approvals Mobile User
S030	SAP Cart Approval Mobile User
S035	SAP Timesheet Mobile User
S040	SAP Sales Order Notification Mobile User
S045	SAP Material Availability Mobile User
S050	SAP ERP Order Status Mobile User
S055	SAP CRM Retail Execution Mobile User
S060	SAP Field Service Mobile User
S065	SAP EAM Work Order Mobile User
S070	SAP ERP Quality Issue Mobile User
S075	SAP Customer and Contacts Mobile User
S080	SAP GRC Access Approver Mobile User
S085	SAP GRC Policy Survey Mobile User
S090	SAP Payment Approvals Mobile User
S095	SAP Customer Financial Fact Sheet Mobile User
S100	SAP Interview Assistant Mobile User

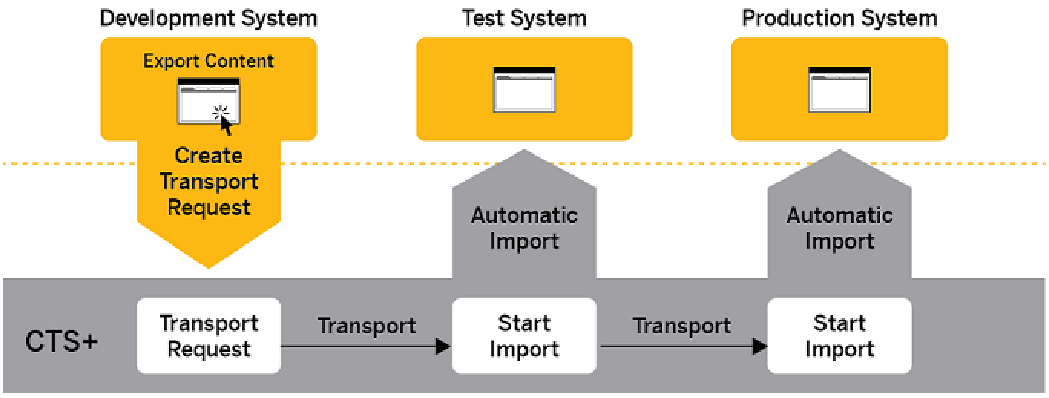
ID	Unit
S105	SAP Transport Notification and Status Mobile User
S110	SAP Transport Tendering Mobile User
S115	SAP Manager Insight Mobile User
S120	SAP Electronic Medical Record Mobile User
S125	SAP Sales Mobile User

CTS Overview

CTS is a transport management system that enables you to distribute artifacts and automate deployment to different target systems that are connected through transport routes.

Administrators can use CTS to distribute Sybase Unwired Platform development artifacts, and automate deployment to different Unwired Servers. For example, administrators can use CTS to transfer Sybase Unwired Platform artifacts from a development Unwired Server to a test Unwired Server, and then to a production Unwired Server.

The SAP Change and Transport System (CTS) of ABAP has been enhanced so that it can be used for transporting non-ABAP objects as well. This system is known as CTS+ or enhanced CTS.



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

Table 21. Additional Resources

Information	Location
Documentation for CTS including CTS Plug-In	On the SAP help portal: *http://help.sap.com/nwcts*
Central note for CTS+	*1003674*
Central note for SL Toolset	*1563579*
Central note for CTS plug-in	*1665940*

Basic Setup for CTS

Before using CTS to transport Unwired Platform artifacts, you need to complete basic CTS setup. You need to create a CTS domain, configure the Transport Organizer Web UI and set up CTS Deploy Web Services.

To use CTS to transport Sybase Unwired Platform development artifacts, your landscape must include components that interact with Unwired Platform components. For a list of these components, see *Supported Hardware and Software*.

Table 22. CTS Basic Setup

Task	Notes
Create a CTS domain.	If your SAP Solution Manager is not set as the CTS Domain Controller, set it using transaction code STMS. Documentation for creating a CTS domain is on the SAP help portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/44/b4a0b47acc11d1899e0000e829fbbd/frameset.htm*

Task	Notes
Configure the Transport Organizer Web UI.	You need to activate services for the Transport Organizer Web UI in order to use it. You need to activate services for the Object List Browser in order to see a detailed list of objects attached to a transport request as part of one file. If CTS is already in use on the SAP Solution Manager where you are doing the configuration, the services should already be activated. Documentation for activating services for the Transport Organizer Web UI is on the SAP help portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/e5/998566c2174196a12b72e7c7af51e7/frameset.htm* **Note:** If you receive error messages when running this application later on, or if you don't want to activate all ICF services, read the error messages carefully and activate the services named in the error messages using transaction SICF.
Configure the CTS Deploy Web Service.	You need to create an RFC connection for the communication between the AS ABAP and the AS Java of the CTS system. Documentation for configuring the CTS Deploy Web Service is on the SAP help portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/2b/326d6274134cea8b217f24889d19c1/frameset.htm* **Note:** CTSDEPLOY is the standard port that should be used to connect the Deploy Web Service client and the Deploy Web Service. If this port is already in use for other CTS scenarios and cannot be used for Unwired Platform, create an additional port with a different name, for example, CTSDEPLOY_SUP.

Configuring CTS

To configure CTS for transporting Sybase Unwired Platform development artifacts between different Sybase Unwired Platform system roles, administrators must create an Unwired

Platform application type, set up systems and transport routes for the Sybase Unwired Platform server environments, and prepare Unwired Platform transport scripts.

Table 23. CTS Configuration Tasks

Task	Notes
Create an application type for Sybase Unwired Platform.	If the Sybase Unwired Platform application type is already defined in CTS+, you do not need to complete this task.
Set up source and target systems for the required server environments.	Documentation for creating systems in the Transport Management System is on the SAP Help Portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/bf/e4626214504be18b2f1abeeaf4f8e4/frameset.htm* When creating your target system, select **Other** as the deployment method.
Define transport routes to connect your systems.	Documentation for configuring transport routes is on the SAP Help Portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/ en/44/b4a1df7acc11d1899e0000e829fbbd/ frameset.htm*
Prepare the Sybase Unwired Platform transport scripts .	

Setting Up the Application Type in CTS

You must set up a Sybase Unwired Platform application type in CTS to connect the Unwired Platform with CTS so you can transport Unwired Platform development artifacts.

1. In the SAP Solution Manager, enter the transaction code `STMS`.
2. In the Transport Management System, click **Overview** > **Systems** (Shift+F6).
3. Choose **Extras** > **Application Types** > **Configure**.
4. Click **Edit** > **New Entries** (F5)..
5. Enter:

Field	Value
Application Identifier	`SUP`

Field	Value
Application Description	A description of what the application type is used for. For example, Sybase Unwired Platform objects.
Support Details	Contact SAP support by creating a customer message using component `MOB-SUP`.

6. Save your entries and confirm that you want to distribute the configuration.

CTS+ is now prepared to handle Sybase Unwired Platform content. You can now select the Sybase Unwired Platform application in the Transport Organizer Web UI when choosing the file object to be attached to a transport request.

Configuring the Target System in CTS

You must configure the Sybase Unwired Platform target system in CTS for transporting Sybase Unwired Platform application objects.

1. In the SAP Solution Manager, enter the transation code `STMS` .
2. In the Transport Management System, click **Overview** > **Systems** (Shift F6).
3. Choose **SAP System** > **Create** > **Non-ABAP System**.

   The TMS: Configure Non-ABAP System dialog displays.
4. Create the non-ABAP system with an appropriate system ID and a description.
5. Select the SAP Solution Manager as the communication system.
6. Under Target System Settings, select **Activate Deployment Service**, and select a deployment method of **Other**.
7. Save your settings and confirm that you want to distribute the TMS configuration.
8. In the next displayed screen, click **Edit** > **New Entries** (F5).
9. Select the **SUP** application type.
10. Specify values:

Field	Description
Deploy Method	Select **Script-based Deployment** .

Field	Description
Deploy URL	Enter the name and port number of the target Unwired Server using this format: `*hostname*:*hostport*` If you are transporting MBO package archives, you can override the domain referenced in the archive by specifying the domain as part of the URL using ths format: `*hostname*:*hostport*?domain#*domainname*` If you are transporting application or Hybrid App archives with the same transport request as an MBO package archive, this information is used only for the MBO package archives. **Note:** Only the management port, which by default is 2000, is supported. The secure management port is not supported.
User	Enter `supAdmin`.
Password	Enter the password for the SUP administrator.

11. Click **Table View > Save**.

Preparing Transport Scripts for CTS

Sybase Unwired Platform archives are imported by CTS using a script-based deployment. Prepare the transport script using `supadmin.bat`.

Prerequisites

Because transport scripts rely on `supadmin.bat`, validate that the installed version of JRE is 1.6+.

Task

1. Download `supadmin.zip` from *`SUP_HOME`*`\Servers\UnwiredServer\bin\CTSPlus`.
2. On the server hosting CTS, check for the existence of a `CtsScripts` subfolder. If it does not exist, create it.

   To determine where to create the `CtsScripts` subfolder, use the System Information Console to find the value of the `sys.global.dir` system property. For example, the default value of the `sys.global.dir` property on Windows platforms is: `C:\usr\sap\<SID>\SYS\global`.
3. Extract `supadmin.zip` into the `CtsScripts` subfolder.
4. Change the JAVA_HOME variable in `deploy_SUP.bat`, to point to a JDK 1.6+ installation on the local server.

   Note: You can search for JAVA_HOME in `deploy_SUP.bat` to find and modify it.

5. Set security settings according to your needs and security policy.
- The CtsScript folder must be executable by the SAP sidadm user of the CTS host system.
- Restrict access to the `CtsScripts` folder to users who are performing setup and deployment.

Transporting Artifacts Between Servers Using CTS

To transport development artifacts between environments using CTS, Sybase Unwired Platform administrators must export the application or package from the development environment, then use CTS to transport the artifacts to other environments. Administrators create a transport request in CTS, attach the export archive, and import the transport request to the testing or production Sybase Unwired Platform system.

Note: You must have set up the Sybase Unwired Platform source and target systems in CTS and included them in a transport route before attempting to transport Sybase Unwired Platform artifacts.

Table 24. Administration Tasks

Task	Perform in
Export the package, application, or Hybrid App.	Sybase Control Center (console or command line)
Create a transport request and attach the package, application, or Hybrid App archive.	CTS
Release the transport request.	CTS See the SAP Help Portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/0e/70a0ed3b6943de93b083496b0f42a0/frameset.htm*
Import the transport request to the target system.	CTS See the SAP Help Portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/4b/b9a1222f504ef2aa523caf6d22d1c9/frameset.htm*

See also
- *MBO Package Export Archive* on page 318
- *Hybrid App Export Archive* on page 319
- *Application Export Archive* on page 319

Exporting Packages, Applications and Hybrid Apps

You can export MBO packages, applications, and Hybrid Apps using either the Sybase Control Center administration console or the command line.

Exporting MBO Packages

Export an MBO package to bundle one or more MBOs and package options to create a new instance of a deployment archive. Use the deployment archive to transport the package between Unwired Servers.

Prerequisites

Before beginning, review import requirements and best practices.

Task

1. In the left navigation pane of Sybase Control Center, expand **Domains >** ***Domain name*** **> Packages**.
2. In the right administration pane, select the box adjacent to the name of the package and click **Export**.
3. Click **Next**.
4. Click **Finish**.
5. Select the file system target for the exported contents and click **Save**.

   Note: Ensure that you do not hide the file type extension when you name the export archive; otherwise, the *.zip extension becomes invisible, which adversely affects the outcome of the export process.

   A status message indicates the success or failure of the export transaction. If the transaction succeeds, a ZIP file is created in the location you specified. You can then import this file on another Unwired Server.

Next

Deliver the file to the appropriate person, or deploy or transport the exported package to the appropriate server.

Exporting Applications

Export applications to create a deployment archive that can be used to transport applications between Unwired Servers.

Prerequisites

Before beginning, review import requirements and best practices.

Task

1. In the left navigation pane of Sybase Control Center, select **Applications**.
2. In the right navigation pane, click the **Applications** tab.
3. Select the box adjacent to the application and click **Export**.
4. Click **Next**.
5. Click **Finish**.
6. Select the file system target for the exported contents and click **Save**.

   Note: Ensure that you do not hide the file type extension when you name the export archive; otherwise, the *.zip extension becomes invisible, which adversely affects the outcome of the export process.

   A status message indicates the success or failure of the export transaction. If the transaction succeeds, a ZIP file is created in the location you specified. You can then import this file on another Unwired Server.

Next

Deliver the file to the appropriate person, or deploy or transport the exported application to the appropriate server.

Exporting Hybrid Apps

Export Hybrid Apps to create a deployment archive that can be used to transport Hybrid Apps between Unwired Servers.

1. In the left navigation pane of Sybase Control Center, select **Hybrid App**s.
2. In the right navigation pane, click the **General** tab.
3. Select the box adjacent to the Hybrid App and click **Export**.
4. Click **Next**.
5. Click **Finish**.
6. Select the file system target for the exported contents and click **Save**.

   Note: Ensure that you do not hide the file type extension when you name the export archive; otherwise, the *.zip extension becomes invisible, which adversely affects the outcome of the export process.

   A status message indicates the success or failure of the export transaction. If the transaction succeeds, a ZIP file is created in the location you specified. You can then import this file on another Unwired Server.

Next

Deliver the file to the appropriate person, or deploy or transport the exported Hybrid App to the appropriate server.

Creating a Transport Request in CTS

Create a transport request in CTS to transport an exported Sybase Unwired Platform object to another Sybase Unwired Platform system and to import it.

Note: For more information on using the Transport Organizer Web UI, see the following topic on the SAP help portal:

http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/df/7a1d1a4f0d4805b46c61a0d53cb4c7/frameset.htm

1. In the SAP Solution Manager, enter the transation code `STMS`.
2. Click **Environment > Transport Organizer Web UI**.
3. In the System dialog box, select the system ID of the source system to create your transport request for.

   The Transport Organizer Web UI opens in the browser.
4. Click **Create Request**.
5. Enter a short description in the Create Request dialog box.
6. (Optional) Enter the project and a different request owner (if necessary) for the transport request.
7. Click **Create**.
8. Select the transport request, choose the **Object List** tab, switch to change mode, and click **Attach**.
9. Select **Client** (to upload files).
10. Browse to the file location where you exported the archive file.
11. Select **SUP** as the application, and click **Add to List**.
12. Repeat for any additional archive files. You can attach multiple archive files to one transport request. The attached filesappear on the **Files to be Attached** tab in the lower half of the screen.
13. When all required files are selected, click **Attach**.
14. The selected files appear on the **Object List** tab in the lower half of the Transport Organizer screen.
15. When all required files are selected, click **Save Changes**.
16. The UI switches back to display mode, and all successfully attached files are displayed in the **Object List** tab in the lower half of the Transport Organizer screen.

You can now release the transport request to export the objects in the transport request. For more information on releasing transport requests, see the following topic on the SAP help portal:

http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/0e/70a0ed3b6943de93b083496b0f42a0/frameset.htm

After a successful export, you can import the transport request. For more information on importing transport requests, see the following topic on the SAP help portal:

http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/4b/b9a1222f504ef2aa523caf6d22d1c9/frameset.htm

Import Requirements and Best Practices

Import is typically used to move a package or application from a development environment to a test environment, and after testing to a production environment.

MBO Package Imports

- **Domain requirements –** all server connections and security configurations referenced by the MBO package must exist in the target domain.
- **Versioning recommendations –** if a developer has updated the package version number:

  1. (Required) Verify that this new package version is added to the application.
  2. (Recommended) Whenever possible, use the update instead of import. Otherwise, delete the existing package first, to remove all runtime data for the package including cached data, registered subscriptions, subscription templates, client log, MBO and operation histories, and registered package users. Delete these items only after serious consideration.

Application Imports

Make sure the target system has resources that match those referenced in the export archive file:

- Domains, security configurations, logical roles assigned to application connection templates, proxy endpoint connections (used by ODP applications)
- If MBO packages are included, the domains, security configurations and connections referenced by the MBO package archives

Hybrid App Imports

If MBO packages are referenced in the export archive file, make sure that the MBO packages have already been deployed to the target system.

Note: If the Hybrid App has matching rules, all matching rule search expressions are imported as regular expression types. Other expresssion types such as `Begins with` or `Equals` are imported as `Regular expression`.

Import Results

After an import, the package or application on the target Unwired Server will have the same configuration as on the source Unwired Server.

For MBO package imports, the imported package will have the same:

- Cache group settings

- Synchronization group settings
- Subscription bulk load timeout
- Assigned applications
- Subscription template settings
- Role mappings

For application imports, the imported application will have the same:

- Application properties (ID, display name, description, and domains)
- Application connection templates and template settings
- Application customization resource bundles
- Application push configurations
- MBO packages and Hybrid Apps (if included in the exported application)

For Hybrid App imports, the imported Hybrid App will have the same:

- General settings
- Context variable key/value pairs
- Matching rules
- Hybrid App template assignments

If the Hybrid App already exists on the target server, it is:

- Replaced, if it is a lower version number. Existing application connection settings are preserved.
- Updated, if it is the same version number.

If the Hybrid App name does not exist, a new Hybrid App is created.

See also

- *Troubleshoot CTS Imports* on page 180

Troubleshoot CTS Imports

When using a CTS transport request to import a package or application, a return code and deployment message is recorded in the CTS detail log.

Table 25. Import MBO Package Return Codes

Return Code	Meaning	Possible Causes
0	The import has been successfully completed.	The MBO package is imported successfully into the target Unwired Server.

Return Code	Meaning	Possible Causes
8	Content errors occurred when importing.	• The imported archive is not a valid MBO package archive. Create a new transport request and import it. • The specified import domain does not exist in the target Unwired Server. Create the required import domain in the target system or specify a different domain, then retry the import. • The target Unwired Server does not have the security configuration used by the package, or the security configuration is not assigned to the specified domain. Create the required security configuration in the target system and retry the import. • The target Unwired Server does not have the server connection definition that is used by the MBO package in the specified domain. Create the required server connection in the target system and retry the import.
12	A tool issue occurred during the import. Resolve the problem and import the same transport request again.	• The target Unwired Server is down. • The deployment method properties are not set correctly in the target system in CTS. For example, the Deploy URI is not correct. • The provided user name or password is incorrect, or the user does not have authorization to perform the import operation.

Table 26. Import Hybrid App Return Codes

Code	Meaning/Action	Possible Causes
0	The import has been successfully completed.	The Hybrid App package is imported successfully into the Unwired Server.
8	Content errors occurred when importing.	The archive is not a valid Hybrid App archive. Create a new transport request and import it.
12	A tool issue occurred during the import. Resolve the problem and import the same transport request again.	• The target Unwired Server is down. • The provided user name or password is ncorrect, or the user does not have authorization to perform the import operation.

Table 27. Import Application Return Codes

Code	Meaning	Return Error
0	The import has been successfully completed.	The application is imported successfully into the target Unwired Server.

Code	Meaning	Return Error
8	Content errors occurred when importing.	• The imported archive is not a valid application archive. Create a new transport request and import it. • The specified import domain does not exist in the target Unwired Server. Create the required import domain in the target system, or specify a different domain, then retry the import. • The archive does not include the MBO package archive used by the application. Create a new transport request and retry the import . • The security configuration in the application connection template does not exist on the target Unwired Server. Create the required security configuration in the target system and retry the import.
12	A tool issue occurred during the import. Resolve the problem and import the same transport request again.	The target Unwired Server is down.
13	A tool issue occurred during the import. Resolve the problem and import the same transport request again.	The provided user name or password is incorrect, or the user does not have authorization to perform the import operation.

Note: If the application archive includes the MBO package archive used by the application, the codes may indicate the failure conditions listed for MBO packages.

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.

- 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 are only present on a particular type of host.

By default, Unwired Platform server components are installed in the `C:\Sybase\UnwiredPlatform` directory.

Table 28. 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 EBF installations upgraded from an earlier version of Unwired Platform.
`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.
sapjco	SAP Java Connector files.
`scc_cert`	Certificate files for Sybase Control Center.

Directory	Description
`Servers`	Unwired Platform server components.
`Servers\MessagingServer`	Messaging 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 an unserved license is updated, copy the new files here.
`sup`*`XX`*`upgrade`	Appears only in installations upgraded from an earlier version of Unwired Platform.
`ThirdParty`	License terms of third-party components included in Sybase Unwired Platform.
`Uninstallers`	Uninstallers for Unwired Platform Runtime components.
`Uninstallers\UnwiredPlatform`	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 29. Sybase Control Center installation subdirectories

Directory	Description
`backup`	Backup files.
`bin`	Scripts to start or stop Sybase Control Center management framework components.
`common`	Files shared by Sybase Control Center components.
`conf`	Configuration files, including security providers for administration logins.
`ldap`	LDAP-related files.
`log`	Log files used by Sybase Control Center and its console plug-ins to capture only management framework events. No Unwired Platform data is captured here, except for administration logins.
`plugins`	Managed resource plug-ins.
`rtlilb`	Runtime library files.
sccRepoPwdChange	Sybase Control Center repository password update files.
`server`	Class and library files used by the management framework server.
`services`	Class and library files for Sybase Control Center services.
`shared`	Shared class and library files.
`templates`	Sybase Control Center service or plug-in template files.

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 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.

Note: Port numbers 5701, 5702, and 5011 should be reserved ports.

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.
HTTP ports	HTTP port number on which Unwired Server listens for: • Data change notification (DCN) events (server authentication). • HTTP channel notification (mutual authentication).	8000 for HTTP 8001 for HTTPS	Configure in Sybase Control Center by selecting **Configuration**, clicking the **Web Container** tab and entering a new DCN port or secure DCN port, as required. See *Configuring Security Profiles* in *Sybase Control Center for Sybase Unwired Platform* 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 selecting **Configuration**. In the **Components** tab, select **Replication** or **Messaging**, click **Properties** 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 you 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 Subscription Properties* in *Sybase Control Center for Sybase Unwired Platform* online help. For messaging payloads, see *Configuring Messaging Properties* in *Sybase Control Center for Sybase Unwired Platform* online help.
Messaging server administration	Port number for the messaging service for Sybase messaging clients.	5001 for administration services	Cannot be changed in Sybase Control Center. Use the `*SUP_HOME*\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: `*SUP_HOME*\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 Sybase Control Center repository database port 8282, 8283 for default Web container ports	• 9999 – default RMI agent port. The port is set in: `*SCC_HOME*\services\RMI\service-config.xml` • 2100 – default JMS messaging service port. The port is set in: `*SCC_HOME*\services\Messaging\service-config.xml` • 3638 – default Sybase Control Center repository database port. The default port is set in: `*SCC_HOME*\services\SccSADataserver\service-config.xml` • 8282, 8283 – default Web container ports. The default ports are set in: `*SCC_HOME*\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 Sybase Control Center `agent.log` for error messages.
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 *Setting Relay Server General Properties* in *Sybase Control Center for Sybase Unwired Platform*.

Port	Description	Default Port	Instructions for Changing
Unwired Platform reserved	Port numbers reserved for internal use by Unwired Platform components	2638 4343 6001 5500 8002 27000	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 CacheDB	The main, or cache, data tier service.	The Sybase Unwired CacheDB service must be started before the Unwired Server service can be started.

Compo-nent	Service	Description	Dependencies
	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.
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: `dbsrvXX.exe` Synchronization on application server: `mlsrvXX.exe`
	Messaging synchronization services	`AdminWebServices.exe`, `ampservice.exe`, `JMSBridge.exe`, `LBManager.exe`, `OBMO.exe`, `OBServiceManager.exe`
	Scale-out node services	Data services: `java.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	`dbsrvXX.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
- *Viewing and Editing EIS Connection Properties* on page 37
- *Chapter 11, EIS Connection Management* on page 35

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.

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.
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.

Name	Description	Supported values
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 `databaseStart-Command`, and `databaseStop-Command`. 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 Prepared-Statement.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 Prepared-Statement.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 pingAndSet-SessionAuth 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

- *Managing Connection Pools for Unwired Server Connections* on page 31

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)*.

This list of properties can be used by all datasource types. Sybase does not document all native endpoint properties. However, you can add native endpoint properties, naming them using this syntax:

```
<NativeConnPropName>=<SupportedValue>
```

Table 30. General connection parameters

Name	Description	Supported values
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.	Not supported. Do not set this parameter or leave it set to 0.
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.	Not supported. Do not set this parameter or leave it set to 0.
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.	Not accessible by the customer. Do not set this parameter or leave it set to 0.

Name	Description	Supported values
Use X509	Unwired Platform sets this property when a client uses an X509 certificate as the login credential.	If an EIS RFC operation is flagged for SSO (user name and password personalization keys selected in the authentication parameters) then Sybase Unwired Platform automatically sets the appropriate properties to use X.509, SSO2, or user name and password SSO credentials. The corresponding properties should not be set by the administrator on the SAP endpoint.
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.	Not supported.
GUI Redirect Host	Identifies which host to redirect the remote graphical user interface to.	Not supported.
GUI Redirect Service	Identifies which service to redirect the remote graphical user interface to.	Not supported.
Remote GUI Start Program	Indicates the program ID of the server that starts the remote graphical user interface.	Not supported.

SAP DOE-C Properties

Configure Sybase 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 Sybase SAP® Data Orchestration Engine Connector 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.
DOE SOAP Timeout	Specifies a timeout window during which unresponsive DOE requests are aborted.	Positive value (in seconds). The default is 420 (7 minutes).

Name	Description	Supported values
DOE Extract Window	Specifies the number of messages allowed in the DOE extract window.	Positive value (in messages). The minimum value is 10. The maximum value is 2000. 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.
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 SSL: • 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.
Listener URL	Specifies the DOE-C server listener URL.	Valid DOE-C listener URL, for example `http://<sup_hostname>:8000/doe/publish`.

Name	Description	Supported values
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.

Proxy Properties

(Applies only to OData SDK Application and REST API applications). Proxy properties identify the application endpoint and the pool size for connections to a Proxy server.

Name	Description	Supported Values
User	Corresponds to the username of the backend system.	
Certificate Alias	Sets the alias for the Unwired Platform keystore entry that contains the X.509 certificate for Unwired Server's SSL peer identity.	Use the alias of a certificate stored in the Unwired Server certificate keystore.

Name	Description	Supported Values
Address	Corresponds to the application end-point provided when registering an application.	Must be a valid application endpoint.
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	Corresponds to the password of the backend system.	
Allow Anonymous Access	While using REST services, you can enable/disable anonymous user access	The default value is False. To enable anonymous user access, set the value to True.
Enable URLRewrite	This is a custom property which is used to enable/disable URL Rewrite while using REST services.	To enable URL rewrite, set the value to True. To disable URL rewrite, set the value to False.
Enable HttpProxy	This is a custom property which is used to connect to an EIS using Http proxy. If you set this property to True, you must configure an HTTP proxy host and port on the server. For more information, see *Configuring Unwired Server to Securely Communicate With an HTTP Proxy in the Sybase Control Center online help.*.	The default value is set to False.

Note:

- In Sybase Control Center, when the application endpoint for a registered application is modified under the **Applications** node, you must manually update the **Address** in the proxy properties of the connection pool. There is a default refresh time of 100000ms after which the updated connection settings get reflected at runtime.
- In Sybase Control Center, in addition to the application endpoint, you must register any URL that is required by an application for a proxy service to enable communication with Unwired Server.
- The application end point should be white-listed only once as a proxy connection. The proxy connection name should be same as the application ID, if an application is registered to be used for referencing the EIS service end point.

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.
authentication-Preemptive	When credentials are available and this property is set to the default of false, this property allows Unwired Server to send the authentication credentials only in response to the receipt of a server message in which the HTTP status is 401 (UNAUTHORIZED) and the WWW-Authenticate header is set. In this case, the message exchange pattern is: request, UNAUTHORIZED response, request with credentials, service response. When set to true and basic credentials are available, this property allows Unwired Server to send the authentication credentials in the original SOAP or REST HTTP request message. The message exchange pattern is: request with credentials, a service response.	False (default) True

Name	Description	Supported Values
Socket Timeout	The socket timeout value controls the maximum time in milliseconds after a web service operation (REST or SOAP) is allowed to wait for a response from the remote system; if the EIS system doesn't respond in that time, the operation fails and the SUP thread is unblocked.	Time in milliseconds (default: 6000). Range of [0 – 2147483647], where 0 is interpreted as infinity.

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.

Apple Push Notification Properties

Apple push notification properties allow iOS users to install client software on their devices.

- **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, and Customization Resource of an application connection template

- **Automatic Registration Enabled –** set to **True** to automatically register the application using a connection template. Be sure you understand the security ramifications of setting this value to true. See *Registering Applications, Devices, and Users* in the *Security* guide.
- **Application Identifier –** the application identifier registered on SCC.
- **Customization Resource Bundles –** the application configuration (customization resource bundles) associated with the application. Values include:
  - A single name, such as `Appmc:1.2.1`, indicates a single customization resource bundle.
  - Blank means no customization resource bundles are assigned.

  Note: Application configuration is only used for OData SDK clients (Android and iOS), and is not used for Hybrid Web Container connections.
- **Domain –** the domain selected for the connection template.

  The domain is not required when automatic registration is enabled.
- **Notification Mode –** the notification mode through which native notifications and payload push notifications to registered devices are delivered:
  - Only native notifications – allows third party applications or EIS to deliver native notifications directly through the HTTP notification channel: BlackBerry (BES), Apple (APNS), or Android (GCM) to the device.
  - Only online/payload push – allows third party applications or EIS to deliver data payload push notifications to an online device.
  - Online/payload push with native notifications – allows both payload and native notifications to be delivered to the device.

  Note: Apple and Google do not recommend payload delivery over their systems:

  - Data may not be delivered.
  - Data is delivered out of sequence.
  - If enabled, the actual payloads must be small.

    For example, GCM makes no guarantees about delivery or order of messages. While you might use this feature to inform an instant messaging application that the user has new messages, you probably would not use it to pass actual messages.

  RIM supports guaranteed delivery, including callbacks to notify Unwired Server that the message was delivered or when it failed. However, this is a different message format. RIM messaging has many limitations including packet size, number of packets for a single user that BES keeps, number of packets BES keeps for all users, and so on. Therefore, BES should also only be used to send the notification, but not to send payloads.

Refer to the appropriate platform documentation (APNS, BES, or GCM) for additional information.

- **Security Configuration –** the security configuration defined for the connection template.

  The security configuration of the application connection template is used to authenticate users when automatic registration is enabled. The user name for authentication can be included in the security configuration, for example, supAdmin@admin. If a security configuration is provided, the server looks for the application connection template according to both the appId and security configuration. If a security configuration is not provided, the server looks for the unique application connection template according to the appId. If there are multiple templates with different security configurations for the same appId, the server reports an exception, as it does not know which template should be used to authenticate the user.
- **Logical Role –** the logical role that users must belong to in order to access the application.
- **Template Priority –** the priority in which the application connection template will be used by the application, if the application has more than one application connection template associated with it.

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
BES Push Listener Port	The listener port for BES notifications. The port is discovered and set by the client, and is read-only on the server.

Property	Description
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 for a client application so it can locate the appropriate Unwired Server synchronization service.

Typically, production client applications connect to the synchronization server via Relay Server or some other third-party intermediary reverse proxy server. In those cases, the settings for the synchronization host, port, and protocol need to use Relay Server property values. For more information on how these properties are used in a synchronization environment, see *Replication*.

- **Activation Code –** (not applicable to replication clients) 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 synchronization server stream parameters that are used to explicitly set client-specific values. After the client application successfully registers with the Unwired Server, it receives the trusted certificate configured in the Server configuration (either the Secure Sync Port Public Certificate or Trusted Relay Server Certificate). If you are using Relay Server, ensure the Trusted Relay Server Certificate property is configured to point to a file that has the server's public certificate.

  You can configure these parameters as one or more`name=value` entries.

  - **trusted_certificates –** the file containing trusted root certificate file.

  - **certificate_name –** the name of the certificate, which is used to verify certificate.
  - **certificate_unit –** the unit, which is used to verify certificate.
  - **certificate_company –** the name of the company issuing the certificate, which is used to verify certificate.

  For more information about certificates, see *Security*.
- **Synchronization Server URL Suffix –** the server URL suffix. For Relay Server, suffixes vary depending on the Web Server used. For example, `/cli/iarelayserver/`*`FarmName`* for Apache, or `ias_relay_server/client/rs_client.dll/`*`FarmName`* for IIS.

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 Hybrid App-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. For IIS, this path is relative. Acceptable values include:
  - For IIS – use`/ias_relay_server/client/rs_client.dll`.
  - For Apache – use `/cli/iasrelayserver`.

  Note: The value used in the client application connection for the URL suffix must match what the administrator configures in the URL suffix. Otherwise, the connection fails. Use the Diagnostic Tool command line utility to test these values. See *Diagnostic Tool Command Line Utility (diagtool.exe) Reference* in *System Administration*.
- **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.

    - 1: Basic information, including errors
    - 2: Some additional details beyond basic
    - 3: Medium amount of information logged
    - 4: Maximum tracing of debugging and timing information
    - 5: Maximum tracing of debugging and timing information (currently same as level 4)
- **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 URL pointing to the EIS.
- **Push Endpoint –** the Sybase Unwired Platform URL used by EIS to forward notifications.

Note: The application end point should be whitelisted only once as a proxy connection. The proxy connection name should be same as the application ID, if an application is registered to be used for referencing the EIS service end point.

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.

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
`-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 `<UnwiredPlatform_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 `SUP_HOME\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\SQLAnywhereXX\BINXX`.

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 *`SUP_HOME`* `\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.

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: A `[LicenseNumber]` value is required for *<ED> <EE>* editions.

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 (client-server licenses)

- ST for Seat license
- OT for Other license
- SS for Standalone seat license
- DT for Development and test license

Examples

- **Update Client and Server Licenses**

  To update the client and server license number from the current value to 1200 in an enterprise edition of Unwired Platform, run:

  ```
  license.bat EE CP 1200
  ```

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.

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
- Replication protocol (default: HTTP)
- Replication 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.
`-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 (supadmin.bat) Utility

Deploy, delete, import, and export application packages using the administration with supadmin.bat and its supported commands. 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
Deploy|Import|Export|Delete commandOptions
```

Deploy Command

Deploy 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. • `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 *deploymentUnitFile-Name*	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.
-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 and Export Commands

Import an existing package or application to Unwired Server using the **import** command. Export a package or application from Unwired Server using the **export** command.

Syntax

Note: You cannot import a package file generated from a previous version of Unwired Platform.

To import an MBO package:

```
supadmin.bat -host hostname -port port -u username -pw password
import -file archive name -d domain name
```

To export an MBO package:

```
supadmin.bat -host hostname -port port -u username -pw password
export -type mbopackage -d domain name -id MBO package name -file
archive name
```

To import a Hybrid App package:

```
supadmin.bat -host hostname -port port -u username -pw password
import -file archive name
```

To export a Hybrid App package:

```
supadmin.bat -host hostname -port port -u username -pw password
export -type hybridapp -id Hybrid App name -file archive name
```

To import an application:

```
supadmin.bat -host hostname -port port -u username -pw password
import -file archive name
```

To export an application:

```
supadmin.bat -host hostname -port port -u username -pw password
export -type application -id application name -file archive name
```

Examples

- **Import example –** This command imports the MBO package "c:/imported.zip" file to the domain "default":

```
supadmin.bat -host test01.sap.com -port 2000 -u supAdmin -pw PWD
 import -d default -file c:\imported.zip
```

- **Export example –** This command exports the MBO package "samplePkg" from the domain "default" to make it available to other domains:

```
supadmin.bat -host test01.sap.com -port 2000 -u supAdmin -pw PWD
 export -type mbopackage -d default -id samplePkg:1.0 -file c:
\exported.zip
```

Usage

The import succeeds when all the dependent resources of the import zip exist on the target server.

A successful package import requires that:

- the package domain exists on the target server
- the security configuration referred to by the package exists in the target domain
- the connection referred to by the package exists in the target domain

A successful application import requires that::

- the domain(s) that the application is assigned to exists on the target server
- the domain(s) application connection template referred to by the application exists on the target server. If the application connection template does not exist on the target server, it is imported. You may need to change the connection and proxy settings to match the target server configuration. If the application connection template aready exists on the target server, the target server template connection and proxy settings are used.
- the security configuration referred to by the application connection template exists in the target domain
- the proxy endpoint with same address referred to by the application connection template exists in the target domain

A Hybrid App import has no dependencies.

The import fails when one or more of the dependent resources of the import.zip is missing on the target server.

In any of these cases an import failure leaves the import action incomplete and the package or application will not appear on Unwired Server.

Delete Command

Delete 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
```

Create or Remove the Windows Service for sampledb Server (sampledb) Utility

Creates or removes the Sybase Unwired SampleDB windows service for the sampledb server. Used on an Unwired Server that was not installed with a development license.

Syntax

```
sampledb.bat [install auto | install manual | remove | start |
stop ]
```

Parameters

- **install auto –** creates the Sybase Unwired SampleDB Windows service in auto-start mode for the sampledb server.
- **install manual –** creates the Sybase Unwired SampleDB Windows service in manual start mode for the sampledb server.
- **remove –** removes the Sybase Unwired SampleDB service.
- **start –** starts the Sybase Unwired SampleDB service.
- **stop –** stops the Sybase Unwired SampleDB service.

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]
[-cn clusterName] [-nv "<propertyName=NewValue>"] [-v]
```

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.

Note: The -r, -x, and -f options are reserved for internal product use only. While these options are supported by the command line, they are not intended for administrator use.

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.

Usage

Before running this utility, ensure that the data tier is available; otherwise platform data is not modified correctly.

See also

- *Changing Database Ports for SQLAnywhere Databases* on page 27
- *Changing SQLAnywhere Database Server Thread Count and User Startup Options* on page 28

Diagnostic Tool Command Line Utility (diagtool.exe) Reference

Ensure a valid configuration from end-to-end using the diagnostic tool (**diagtool**) command line utility.

Default file location: The **diagtool.exe** tool is located in the Sybase Unwired Platform installation folder at `C:\Sybase\UnwiredPlatform\Servers\UnwiredServer\diagtool`.

Note: You can create a zip file of the entire contents of the folder and then run the **diagtool.exe** command after extracting the zip file. The diagtool is a self-sufficient executable that can run on any Windows machine.

Syntax

```
diagtool verify options
```

Parameters

This command line utility can be used to validate different aspects of Unwired Platform. Parameters are organized by use cases it supports:

Table 31. Parameters for Command Support

Parameter	Description
verify	Verify specific configuration

Table 32. Parameters for Supported Sub-commands

Parameter	Description
-push	Verify messaging connection
-sync	Verify synchronization connection
-register	Verify automatic registration configuration for a specified application
-security	Validate specific security configuration
-e2ee	Validate end-to-end security configuration setup

Table 33. Parameters for Common Connection Options (Required for All Sub-commands)

Parameter	Description
-h	Host name of the server (default is the local machine host name)
-P	Messaging server port (default value is 5011)
-u	User name that is validated by the specified security configuration or "admin" security configuration per the verification command
-p	Password

Table 34. Parameters for Messaging Verification

Parameter	Description
-push	Validate messaging connection

Table 35. Parameters for Security Configuration Verification

Parameter	Description
-security	Verify security configuration
-sc	Security configuration

Table 36. Parameters for Messaging Synchronization Verification (Relay Server and HTTPS)

Parameter	Description
-urlsuffix	Relay Server URL suffix for the messaging farm
-farmid	Relay Server farm ID for the messaging farm
-https	(Optional) Use HTTPS protocol for the messaging connection (-P should be the HTTPS port)
-cert	(Optional) Specify the messaging server certificate (only needed if using an unknown or self-signed certificate)

Table 37. Parameters for Diagtool Verbosity Option

Parameter	Description
-verbose	(Optional) Output detail trace information

Table 38. Parameters for End-to-End Encryption Verification

Parameter	Description
-e2ee	Validate end-to-end encryption

Table 39. Parameters for Replication Synchronization Verification

Parameter	Description
-sync	Verify synchronization
-synchost	Specify the synchronization host name (of the Unwired Platform server or Relay Server of the intermediary load balancer)
-syncport	Specify the synchronization port number (of the Unwired Platform server or Relay Server or the intermediary load balancer)
-syncfarmid	Relay Server farm ID for the synchronization

Parameter	Description
-syncurlsuffix	Relay Server URL suffix for the synchronization farm
-synchttps	(Optional) Use HTTPS for synchronization connection (-syncport should be the HTTPS port)
-synccert	(Optional) Full path to the synchronization server certificate (only needed if using an unknown or self-signed certificate)

Table 40. Parameters for Automatic Registration Verification

Parameter	Description
-register	Verify automatic registration
-appid	Application ID for which the verification must be done (security configuration can be optionally specified)

Note: The "admin" security configuration is used for validating credentials passed in to verify messaging and synchronization connection. Similarly, the "admin" security configuration is used by default for the automatic registration command when not specified.

Note: The diagnostic tool does not support validation of certificate login module based security configuration.

Examples

- **Verifying security configuration –** Verifies client-side security configuration.

```
diagtool verify -security -h host -P port -u username -p password
-sc security_configuration
```

- **Verifying automatic registration without Relay Server –** Verifies automatic registration in the configuration.

```
diagtool verify -register -appid application_ID -h host -P 2480 -
u username -p password
```

- **Verifying encryption –** Verifies encrypted configuration.

```
diagtool verify -e2ee -h host -P port -u username -p password -
farmid farm_ID -urlsuffix URL_suffix -synchost sync_hostname -
syncport sync_port_number
```

For more information, see *Syntax and Scenarios*.

Diagnostic Tool Command Line Utility

Ensure a valid client-side configuration from end-to-end in the Sybase Unwired Platform landscape by using the diagnostic tool command line utility (**diagtool**).

Use the **diagtool** utility to verify a valid end-to-end configuration. The tool runs on Windows from the command line. This utility verifies various aspects of the server configuration after doing a post-installation configuration. Use the utility to verify:

- changes to an existing or new security configuration
- connectivity to the push and sync ports of the Sybase Unwired Platform server
- connectivity to those ports via Relay Server and outbound enablers
- transport level security
- end-to-end encryption configuration on the server
- that a given application is properly configured for automatic registration-based on-boarding of the application users

Upon execution of the diagnostic tool command, the result is success or failure with a root cause which administrators can use to identify the potential issue at a fine-grained level. This tool eliminates the need to build an application or a verification tool to verify the supported configuration verification. Additionally, administrators can use **-verbose** option to get detailed information on the sequence of steps to perform the selected configuration test. It is important that the administrator performs the tests in a specific order.

For more information, see *Diagnostic Tool Command Line Utility (diagtool.exe) Reference* and *Syntax and Scenarios*. Also see *Command Line Utilities*.

See also

- *Syntax and Scenarios* on page 241

Syntax and Scenarios

Use these diagnostic tool scenarios to run verification tests for client-side configurations.

- **Scenario: Verify security configuration** - Verify that a security configuration is properly configured on Unwired Server.

  ```
  diagtool verify -security -h host -P port -u username -p password
  -sc security_configuration
  ```
- **Scenario: Verify connection to Unwired Server messaging port** - Verify that the messaging client is connecting to Unwired Server directly.

  ```
  diagtool verify -push -h host -P port -u username -p password
  ```
- **Scenario: Verify the connection to Sybase Unwired Server messaging port via a Relay Server** - Verify the messaging client connection to Unwired Server via Relay Server.

  ```
  diagtool verify -push -h host -P port -u username -p password -
  farmid farm_ID -urlsuffix URL_suffix
  ```

- **Scenario: Verify connection to Unwired Server replication synchronization port** - Verify that the replication client is connecting to Unwired Server directly.

```
diagtool verify -sync -h host -P port -u username -p password  -
synchost sync_host -syncport sync_port
```

- **Scenario: Verify connection to Unwired Server via Relay Server to both messaging and replication ports** - Verify a connection to Unwired Server via Relay Server.

```
diagtool verify -sync -h host -P port -u username -p password -
farmid farm_ID -urlsuffix URL_suffix -synchost sync_host -
syncport sync_port -syncfarmid sync_farm_ID -syncurlsuffix
sync_URL_suffix
```

- **Scenario: Verify automatic registration** - Verify that automatic registration is set up for the application ID in Unwired Server.

```
diagtool verify -register -appid application_ID -h host -P port -
u username -p password
```

- **Scenario: Verify automatic registration via Relay Server** - Verify that automatic registration is set up for the application ID in Unwired Server connecting via Relay Server.

```
diagtool verify -register -appid application_ID -h host -P port -
u username -p password-farmID farm_ID -urlsuffix URL_suffix
```

- **Scenario: Verify Encryption Security** - Verify that end-to-end encryption security is set up in Unwired Server for a replication sychronization client connecting via Relay Server.

```
diagtool verify -e2ee -h host -P port -u username -p password -
farmid farm_ID -urlsuffix URL_suffix -synchost sync_host -
syncport sync_port -syncfarmid sync_farm_ID -syncurlsuffix
sync_URL_suffix
```

Note: To test a messaging connection via a secure port of Relay Server, add the `-https` option to the command line. Similary, add `-synchttps` to test a connection via the secure port of Relay Server for replication synchronization connection. You may also use the `-cert` option to specify the location of the file containing trusted certificates. If you are using a third-party proxy or load balancer between the client and Unwired Server instead of Relay Server, you will use the direct connection option to specify the load balancer host and port information.

See also

- *Diagnostic Tool Command Line Utility* on page 241

DOE-C Utilities

To perform tasks needed when working with 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 `SUP_HOME\MobileSDK\ObjectAPI\Utils\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 *SUP_HOME*`\MobileSDK\ObjectAPI\Utils\bin`.

Syntax

```
codegen -android|-cs|-oc -client -doe -sqlite|-ulj
[-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.
- **-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.
- **-output –** specifies the output directory where generated code is placed. Defaults to `SUP_HOME\MobileSDK\ObjectAPI\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 for Sybase Unwired Platform > Deploy > DOE-C 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 for Sybase Unwired Platform > Administer > Security Configurations*.

Table 41. 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`.

Symbols	Meaning	Example
{...}	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 *SUP_HOME*\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 246.
- 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="DOECUser" />
    </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="optionName1" 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 `SUP_HOME\Servers\UnwiredServer\doe-c_clu\bin`.
2. Enter:

   ```
   execute-commandXMLFile -f 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 `SUP_HOME\Servers\UnwiredServer\doe-c_clu\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 42. Administrative commands

Operation	Command
Start Command Line Utility console to enter commands interactively	`SUP_HOME\Servers\UnwiredServer\doe-c_clu\bin\CLU.bat` (from a command prompt)
Run Command Line Utility to take commands from an XML file	`SUP_HOME\Servers\UnwiredServer\doe-c_clu\bin\execute-commandXMLFile.bat -f 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 43. 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

Operation	Command

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]`

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 44. 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]`

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]`

Operation	Command
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 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.

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

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

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

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.

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 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 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 271

- *Relay Server Configuration Files* on page 274
- *Data Tier Configuration Files* on page 287

Admin Security (default.xml) Configuration File Reference

Defines authentication and attribution properties for the 'admin' security configuration. The file is located in `SUP_HOME\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 Attribute Values* 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://<hostname>: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 ActiveDirectory when you set the password attribute.
InitialContextFactory	None	The factory class used to obtain initial directory context.

Property	**Default**	**Description**
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.
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: `"(&({cer-tattr}={0})(ob-jectclass=per-son))"`	The certificate authentication search filter. The filter may contain the special string `"{cer-tattr}"` which is replaced with the certificate attribute or the map-ped LDAP attribute (if mapping be-tween 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 build-ing 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 ex-ample, `<config:options name="BindPassword" encrypted="true" val-ue="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: `(&(object-class=ldapsuben-try) (object-class=nsroledefini-tion))` For Netscape Directory Server: `(object-class=groupof-names) (object-class=groupofuni-quenames))` For ActiveDirectory: `(ob-jectclass=groupof-names) (object-class=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.
UserFreeformRoleMember-shipAttributes	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 control flag (controlFlag) attribute values and definitions as those defined in the JAAS specification.

If you stack multiple providers, you must set the control flag attribute for each enabled provider.

Control Flag Value	Description
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 (default)	The LoginModule is not required. Regardless 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 user name and 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 succeeds, Sybase Unwired Platform picks up roles based on the Windows groups they are in.

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 265
- *Relay Server Configuration Files* on page 274
- *Data Tier Configuration Files* on page 287

Sybase Control Center Services (service-config.xml) Configuration Files

Configure properties for various Sybase Control Center services. These files are located in *SCC_HOME*`\services\`*<serviceName>*.

Key `service-config.xml` files include:

File	Description	Defaults
SCC_HOME`\services\RMI\service-config.xml`	Sets the RMI agent port.	RMI port: 9999
SCC_HOME`\services\Messaging\service-config.xml`	Sets the JMS messaging service port.	JMS messaging port: 2100
SCC_HOME`\services\SccSADataserver\service-config.xml`	Sets the Sybase Control Center repository database port, and other properties.	Repository database port: 3683
SCC_HOME`\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 *SCC_HOME*`\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.
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.

Property	Default	Description
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	`SCC_HOME\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	${com.sybase.ua.home}/log/agent.log	The name and location of the Sybase Control Center log file.

Property	Default	Description
log4j.appender.agent.MaxFileSize	25MB	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 265
- *Sybase Control Center Configuration Files* on page 271
- *Data Tier Configuration Files* on page 287

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.

  Specify a value and a unit (units are KB, MB, or GB); for example 2048 GB. If you do not specify a value, the default is 10MB. Sybase recommends adding the unit to the value to clarify the configuration setting.
- **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
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

Attribute	Value	Required
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
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

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>
```

Data Tier Configuration Files

Use the various configuration files of the data tier to control behavivior of the databases used by Sybase Unwired Platform.

See also

- *Unwired Server Configuration Files* on page 265
- *Sybase Control Center Configuration Files* on page 271
- *Relay Server Configuration Files* on page 274

Cache Database Startup Options (cdboptions.ini) Initialization File

These options control how the cache server and CDB is started.

If you use `updateprops.bat` to set options for the cache, you aslo need to set them here if you need these changes to be permanent. Otherwise, Sybase recommends that you use sup.properties and set the cdb.user.options propety which automatically sets values in this INI file.

The default startup options table documents which options are set by default. You cannot set or change these values. You can only add new options and set custom values for them. For details on what new options you can set, see *Database Server Options* topics at *http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.sqlanywhere.12.0.1/dbadmin/server-database-dbengine.html*.

Table 45. Default Startup Options

Option	Description
`-n` *`serverName`*`_primary`	Database receives the name of the database file with the path and extension removed.
`-ti 0 -c 24M`	Disconnects connections that haven't submitted a request in 24 minutes.
`-gn 300`	Sets 300 as the maximum number of active tasks for both the cache database server.
`-xf "U:\target\bin\..\mirror-state.txt"`	Specifies the location of the file used for maintaining state information about your database mirroring system. This option is only used in the command to start the arbiter server in a database mirroring system.
`-x tcpip(PORT=5200)`	Specifies server-side network communications protocols, in this case, TCP/IP on port 5200.

Option	Description
`-o "U:\target\bin\..\logs\errorlog.txt"`	Prints all database server messages to the database server error log file.

See also

- *Changing Database Ports for SQLAnywhere Databases* on page 27
- *Changing SQLAnywhere Database Server Thread Count and User Startup Options* on page 28

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 94

mms_rbs_request Table

Detailed history information for replication-based synchronization.

Column name	Column type	Description
id	integer	The identifier of the data row.
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.

Column name	Column type	Description
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.
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.

Column name	Column type	Description
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.
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.

Column name	Column type	Description
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.
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.

Column name	Column type	Description
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.
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.
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.

Column name	Column type	Description
messageType	varchar(50)	The type of message. One of: SUBSCRIBE, UNSUB-SCRIBE, OPERATION_RE-PLAY, RECOVER, SUS-PEND, RESUME, RESUME-NOREPLAY, IMPORT_DATA, DATA_RESET, LOGIN, UN-KNOWN_TYPE.
isError	bit	Value is 1 if errors were recor-ded during transaction. 0 if no errors recorded.
payloadSize	integer	The size of the message pay-load.
isPushMsg	bit	Value is 1 if the message is out-bound, 1 if otherwise.
isRequestMsg	bit	Value is 1 if the message is in-bound, 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 re-play, 0 if not.
sentPayloadSize	integer	The payload size of the out-bound message.
receivedPayloadSize	integer	The payload size of the inbound message.
isMonitored	bit	Value is 1 if MBO is monitored, 0 if not.
isLogged	bit	Value is 1 if the domain is log-ging data, 0 if not.
applicationId	varchar(100)	The identifier for the applica-tion 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.

Column name	Column type	Description
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.

Column name	Column type	Description
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.
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
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.

Column	Column type	Description
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.
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_maxfullrefereshtime` `get_package_mbo_minfullrefereshtime`	• @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_averageachewaittime` `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.
`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.

Domain Log Categories

Domain log data provides detailed statistics 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

Synchronization logs include data related to different aspects of data synchronization, including data and Unwired Server interface.

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.

- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Operation Replay

Synchronization logs include data related to different aspects of data synchronization, including operations and Unwired Server interface.

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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Subscription

Synchronization logs include data related to different aspects of data synchronization, including subscriptions and Unwired Server interface.

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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Result Checker

Synchronization logs include data related to different aspects of data synchronization, including result checker and Unwired Server interface.

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.

- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- None

Cache Refresh

Synchronization logs include data related to different aspects of data synchronization, including cache refresh and Unwired Server interface.

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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Refresh Type
- Virtual Table Name
- Partition Key

- Pre Update Cache Image (payload)
- Post Update Cache Image (payload)

DS Interface

Synchronization logs include data related to different aspects of data synchronization, including data service and Unwired Server 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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

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 Hybrid App DCN.

General Data Change Notification

Provides logging data for general data change notifications between an enterprise information system (EIS) and an MBO package.

- 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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Hybrid App Data Change Notification

Provides logging data for Hybrid App data change notifications between an enterprise information system (EIS) and an MBO package.

- Time – the time and date stamp for the log entry.
- Hybrid App ID – the unique identifier associated with a Hybrid App.
- 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 Hybrid App DCN request subject line.
- From – the "From" value for the Hybrid App DCN request.
- To – the "To" value for the Hybrid App DCN request.
- Body – the message body for the Hybrid App DCN request.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

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. Check the detail pane for additional columns that may be available. Enable Payload to see payload data that may be available.

DOE Connection

Connections log data includes domain connections for DOE connection types, if enabled. Check the detail pane for additional columns that may be available. Enable Payload to see payload data that may be available.

- 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 subscription user for the package.
- Event Type – the DOE-C event type, such as Acknowledged, Duplicate Ignored, Exclude, No Response (from client or server), Packet Dropped, Registration Response, Resend (from client), Status Request (from client or server), DOE-C Subscription, and DOE-C Data Import.
- 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. Its value is DOE for DOE-C logs.
- 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:

- Device ID – the core and administrative (MMS) device ID.
- Domain – the core and administrative (MMS) domain name.
- JSON Message Content – the messaging synchronization JSON message (payload). This is the SUP-specific representation of the incoming DOE XML message in JSON format. DOE-C receives XML the payload from DOE in response, which is then parsed and converted to a JSON string and sent to the client.
- XML Message Content – the DOE SOAP messages (payload). This represents either an XML request in a format for sending to DOE by DOE-C, or an XML payload response received from DOE as applicable.

- Endpoint Name – the core and administrative (MMS) endpoint name.
- DOE server message ID – the SAP DOE reliable messaging server message ID.
- DOE client message ID – the SAP DOE reliable messaging client message ID.
- DOE-C server message ID – the DOE-C client-side SAP DOE reliable messaging server message ID.
- DOE-C client message ID – the DOE-C client-side SAP DOE reliable messaging client message ID.
- DOE-C method name – the DOE-C method being executed.
- DOE-C action name – the DOE SOAP action.
- Push to – the messaging asynchronous response queue.
- Address – the remote URL of the DOE server for this subscription (for example, `http://`*`saphost`*`:50015/sap/bc/DOE_ESDMA_SOAP?sap-client=600`).
- Log – the DOE-C subscription-specific log level.
- Extract Window – the DOE extract window for a subscription. This value determines the maximum number of unacknowledged "in-flight" messages allowed by the DOE reliable messaging protocol.
- PBI – the messaging synchronization "piggy backed import" setting for the subscription.
- Boolean property – indicates whether replay after-images can be piggy-backed onto `replayResult` and `replayFailed` messages (default is false).

JDBC Connection

Connections log data includes domain connections for JDBC connection types to backend data sources, if enabled. Check the detail pane for additional columns that may be available. Enable Payload to see payload data that may be available.

- 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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.

- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Payload and detail columns:

- Input Parameters – the parameters used in a JDBC endpoint operation (payload). This will vary by operation.
- Query – the SQL statement used in a JDBC endpoint operation (payload). This will vary by operation.
- Device ID – the core and administrative (MMS) device ID.
- Domain – the core and administrative (MMS) domain name.
- Endpoint Name – the core and administrative (MMS) endpoint name.
- Database Product Name – the remote database product name, such as "SQL Anywhere".
- Database Product Version – the remote database version, such as "11.0.1.2044".
- Driver Name – the database driver used, such as: "jConnect™ for JDBC™".
- Driver Version – the database driver version, such as "jConnect™ for JDBC™/7.07 GA(Build 26666)/P/EBF19485/JDK 1.6.0/jdbcmain/Wed Aug 31 03:14:04 PDT 2011".
- Database User Name – the database user account.

REST Connection

Connections log data includes domain connections for REST connection types to backend data sources, if enabled. Check the detail pane for additional columns that may be available. Enable Payload to see payload data that may be available.

- 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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.

- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Payload and detail columns:

- Response – the message returned by the EIS system in response to a request (payload).
- Device ID – the core and administrative (MMS) device ID.
- Domain – the core and administrative (MMS) domain name.
- Endpoint Name – the core and administrative (MMS) endpoint name.
- HTTP Header Parameters – "Accept-Encoding: gzip, Accept-Encoding: compress".

SAP Connection

Connections log data includes domain connections for SAP connection types to backend data sources, if enabled. Check the detail pane for additional columns that may be available. Enable Payload to see payload data that may be available.

- 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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Payload and detail columns:

- Parameters – input that was supplied to the operation; this will vary per request and operation (payload).
- Device ID – the core and administrative (MMS) device ID.
- Domain – the core and administrative (MMS) domain name.
- Endpoint Name – the core and administrative (MMS) endpoint name.

- SAP Host – the remote system hostname (if available).
- SAP User – the SAP user for the operation.

SOAP Connection

Connections log data includes domain connections for SOAP connection types to backend data sources, if enabled. Check the detail pane for additional columns that may be available. Enable Payload to see payload data that may be available.

- 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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Payload and detail columns:

- Request (payload) – SOAP messages sent to the remote SOAP service.
- Response (payload) – SOAP messages received from the remote SOAP service.
- Device ID – the core and administrative (MMS) device ID.
- Domain – the core and administrative (MMS) domain name.
- Endpoint Name – the core and administrative (MMS) endpoint name.
- Connection Timeout – the response timeout window, in milliseconds.
- Authentication Type – the authentication type, either "None", "Basic", "SSO2", or "X509".

Push Log

Push logs include log data for all push notifications.

- 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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Device Type – device type from which the push message originated.
- Notification Type – type of notification. For example Native.
- Received Time – a time stamp indicating when the message was received by the server.
- Processing Started Time – a time stamp indicating when the server started processing the message.

Proxy Log

Proxy logs all data made to and 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.
- Response Code – the response status code for the invocation.
- 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.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Post Data
- Request Header Fields
- Response Body
- Response Header Fields

Server Log

Server logs include logging data for 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.
- Package – the name of the package to which the subscription belongs.
- Correlation ID – the unique id associated with the correlated data. see *Correlating Log Data Across Subsystems* for details. For example, root context id and Transaction id appear in every trace entry, and contain the values used to correlate trace entries from different subsystems (buckets).
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Bucket – the subsystem from which the logging data originates. For example Security or Data Services (DS).
- Category – the category or type of information under which the data is logged.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Dispatcher Log

Dispatcher log data includes various dispatcher specific messages, including Messaging and Service.

Messaging Log

Messaging log data includes data for all message-based application data routed by the dispatcher.

- 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.
- Request URL - the Gateway URL.
- Response Code – the response status code for the invocation.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Request Header Fields - the HTTP request header field contained in the application. For example, used by REST API-based application to create an application connection.
- Response Header Fields - the HTTP response header field communicated to the device from the server.

Service Log

Service log data includes application service data routed by the dispatcher.

- 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.
- Response Code – the response status code for the invocation.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.

- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Request Header Fields - the HTTP request header field contained in the application. For example, used by REST API-based application to create an application connection.
- Response Header Fields - the HTTP response header field communicated to the device from the server.

Application Log

Application log data includes application specific messages, including Registration and Setting.

Registration Log

Registration log data includes registration-related application data. The registration log applies only to applications registered through the HTTP channel.

- 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.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Auto Registraton - the automatic connection registration setting of the application, as determined by the `autoreghint` provisioning property.
- Security Configuration - the security configuration assigned to the application.
- Template - the application template used by the application.

Setting Log

Setting log data includes application settings information.

- 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.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Operation – the MBO operation.
- Request Body - the application's HTTP request body field.
- Response Body - the server's HTTP response body field.

Transport Archives

Exporting a package or application creates an archive file, which is used to transport development artifacts between servers.

MBO Package Export Archive

The archive file created when you export an MBO package contains deployment information, third-party libraries, and package properties and settings..

- Deployment information:
  - MBO model
  - Endpoint reference
- Third-party libaries:
  - Third-party JAR files
  - Classes the deployment unit uses
- Package properties and settings:
  - Domain name
  - Package name, type, status, and version
  - Security configuration
  - Subscription bulk-load timeout
  - Role mappings
  - Cache group settings
  - Synchronization group settings
  - Subscription template settings
  - Assigned application IDs
  - Server connection

See also

- *Transporting Artifacts Between Servers Using CTS* on page 175

Hybrid App Export Archive

The archive file created when you export a Hybrid App contains Hybrid App settings, context variables, matching rules, and application connection templates.

- The Hybrid App deployment file.
- The general settings of the Hybrid App, including display name, display icon, and description.
- The context variable key-value pairs for the Hybrid App.
- The matching rules for the Hybrid App.

  Note: All matching rule search expressions are exported as regular expression types. Other expresssion types such as `Begins with` or `Equals`. are exported as `Regular expression`.
- Information about the application connection templates that the Hybrid App is assigned to.

See also

- *Transporting Artifacts Between Servers Using CTS* on page 175

Application Export Archive

The archive file created when you export an application contains application metadata, and all application connection templates, customization resource bundles, MBO packages, Hybrid Apps, and native push notifications used by the application.

- Application metadata, such as archive type, archive version, application ID, application display name, application description, and the domains the application belongs to.
- All application connection templates related to the application. Each application connection template is stored as a separate XML file. The application connection template settings are stored as key-value pairs in the file.
- All customization resource bundle ZIP files used by the application.
- All MBO packages used by the application.
- All Hybrid Apps related to the application.
- All native push notification configurations used by the application.

See also

- *Transporting Artifacts Between Servers Using CTS* on page 175

Index