SYBASE®

System Administration

Sybase Unwired Platform 1.5.2

DOCUMENT ID: DC01205-01-0152-01
LAST REVISED: July 2010

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

Contents

CHAPTER 1 Documentation Road Map for Unwired Platform

Learn more about Sybase® Unwired Platform documentation.

Table 1. Unwired Platform documentation

Document	Description
Sybase Unwired Platform Installation Guide	Describes how to install or upgrade Sybase Unwired Platform. Check the *Sybase Unwired Platform Release Bulletin* for additional information and corrections. Audience: IT installation team, training team, system administrators involved in planning, and any user installing the system. Use: during the planning and installation phase.
Sybase Unwired Platform Release Bulletin	Provides information about known issues, and updates. The document is updated periodically. Audience: IT installation team, training team, system administrators involved in planning, and any user who needs up-to-date information. Use: during the planning and installation phase, and throughout the product life cycle.
New Features	Describes new or updated features. Audience: all users. Use: any time to learn what is available.
Fundamentals	Describes basic mobility concepts and how Sybase Unwired Platform enables you design mobility solutions. Audience: all users. Use: during the planning and installation phase, or any time for reference.

Document	Description
System Administration	Describes how to plan, configure, manage, and monitor Sybase Unwired Platform. Use with the *Sybase Control Center for Sybase Unwired Platform* online documentation. Audience: installation team, test team, system administrators responsible for managing and monitoring Sybase Unwired Platform, and for provisioning device clients. Use: during the installation phase, implementation phase, and for ongoing operation, maintenance, and administration of Sybase Unwired Platform.
Sybase Control Center for Sybase Unwired Platform	Describes how to use the Sybase Control Center administration console to configure, manage and monitor Sybase Unwired Platform. The online documentation is available when you launch the console (**Start > Sybase > Sybase Control Center**, and select the question mark symbol in the top right quadrant of the screen). Audience: system administrators responsible for managing and monitoring Sybase Unwired Platform, and system administrators responsible for provisioning device clients. Use: for ongoing operation, administration, and maintenance of the system.
Troubleshooting	Provides information for troubleshooting, solving, or reporting problems. Audience: IT staff responsible for keeping Sybase Unwired Platform running, developers, and system administrators. Use: during installation and implementation, development and deployment, and ongoing maintenance.

Document	Description

Getting started tutorials	Tutorials for trying out basic development functionality.
	Audience: new developers, or any interested user.
	Use: after installation.
	• Learn mobile business object (MBO) basics, and create a mobile device application:
	  • *Tutorial: Mobile Business Object Development*
	  • *Tutorial: BlackBerry Application Development using Device Application Designer*
	  • *Tutorial: Windows Mobile Device Application Development using Device Application Designer*
	• Create native mobile device applications:
	  • *Tutorial: BlackBerry Application Development using Custom Development*
	  • *Tutorial: iPhone Application Development using Custom Development*
	  • *Tutorial: Windows Mobile Application Development using Custom Development*
	• Create a mobile workflow package:
	  • *Tutorial: Mobile Workflow Package Development*

Sybase Unwired WorkSpace – Mobile Business Object Development	Online help for developing MBOs. Audience: new and experienced developers. Use: after system installation.
Sybase Unwired WorkSpace – Device Application Development	Online help for developing device applications. Audience: new and experienced developers. Use: after system installation.

Document	Description
Developer references for device application customization	Information for client-side custom coding using the client object APIs. Audience: experienced developers. Use: to custom code client-side applications. • *Developer Reference for BlackBerry* • *Developer Reference for iPhone* • *Developer Reference for Mobile Workflow Packages* • *Developer Reference for Windows and Windows Mobile*
Developer reference for Unwired Server side customization – *Reference: Custom Development for Unwired Server*	Information for custom coding using server APIs. Audience: experienced developers. Use: to customize and automate server-side implementations for device applications, and administration, such as data handling. Dependencies: Use with *Fundamentals* and *Sybase Unwired WorkSpace – Mobile Business Object Development*.
Developer reference for system administration customization – *Reference: Administration APIs*	Information for custom coding using administration APIs. Audience: experienced developers. Use: to customize and automate administration at a coding level. Dependencies: Use with *Fundamentals* and *System Administration*.

CHAPTER 2 Introduction to Sybase Unwired Platform

Unwired Platform provides an integrated solution for mobilizing your enterprise applications to multiple device platforms using the same infrastructure and development investment.

Unwired Platform covers end-to-end aspects of mobilizing an application—providing standards-based access to data sources, building heterogeneous device applications with reusable object-oriented frameworks, supporting end-to-end security, and managing deployment, devices, users, and applications.

This guide is to familiarize you with the platform, provide an overview of the platform capabilities, and show how the platform addresses the technical challenges of building and managing mobile enterprise applications.

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

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

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

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

  Before you begin, know: the devices you need to support, the back-ends you need to integrate with, and the synchronization model you will use.
- Mobile application administration requires both the development and deployment of applications. The Unwired Platform perspective in Sybase Control Center is integral to configuring and managing applications as they are developed and integrated into your system landscape. Further, all aspects of a production environment can be monitored for troubleshooting or performance tuning purposes.

  Before you begin, decide: the system design/topology you environment requires and what type of security providers you need to delegate application security to.
- Mobile user, device, and application management simplifies how the end user is registered and provisioned in the Unwired Platform environment. When Afaria® is used in conjunction with Unwired Platform, an administer 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.
  - Afaria manages all aspects of device provisioning for all supported device types, with the exception of iPhone (which requires the use of its proprietary mechanisms).

  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).

See also

- *Chapter 7, System Administration* on page 125

Platform Components

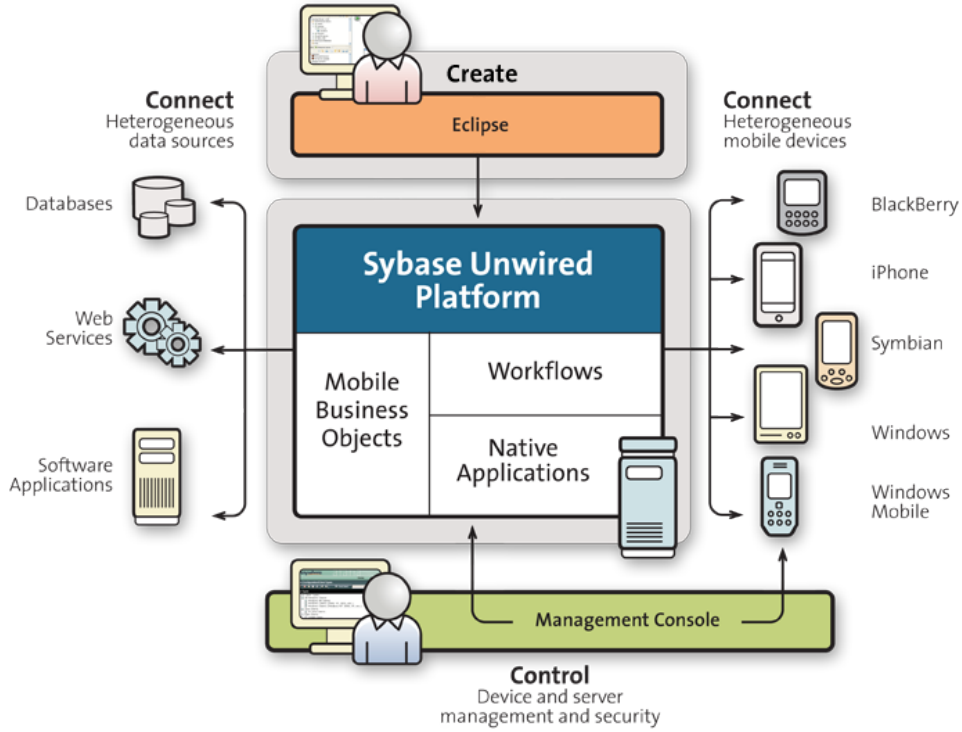
The Unwired Platform architecture consists of a server tier, a data tier, and client tier. In a production environment, the server and data tiers must be installed on 64-bit hosts.

Server-tier components integrate with back-end enterprise systems, data access and transaction services, device and application deployment, and system management functionality. The data-tier is used to store data retrieved from the backend data sources and other runtime related metadata. The client tier consists of device applications built on top of the Unwired Platform client runtime.

You can employ different secure application communication styles—replication and messaging—between the client and the server tiers. Unwired Platform uses a relay server as a component of the enterprise demilitarized zone (DMZ), to securely integrate mobile devices into your system landscape. You need not install any Unwired Platform components on the carrier network or outside your firewall; they can all be installed and controlled within your corporate networks.

For developers, Unwired Platform includes a complete application development environment. Unwired Platform offers a choice of multiple application models—native application development, Device Application Designer development, or simple codeless business process mobilization—by reusing your development and deployment investments in the Unwired Platform server tier. If you choose, you can install the Sybase Unwired Platform Windows Mobile .NET Component in your Microsoft® Visual Studio installation. See *Application Development Environment*.

Figure 1: Sybase Unwired Platform



Unwired Server

The Unwired Platform runtime server is called Unwired Server. Unwired Server manages the data exchange process between the enterprise and device clients to create a homogeneous

layer in a diverse mobile ecosystem. In a production environment, the Unwired Server must be installed on a 64-bit host.

Unwired Server features include:

- Data services – supports connections to back-end data resources using these standard technologies: enterprise databases with JDBC connections and Web Services (SOAP-style and REST-style) . Also supports connections to enterprise applications such as SAP® and Remedy®.
- Data virtualization – introduces a layer called a mobile business object (MBO) between your enterprise databases or applications, and the remote database on the device client. Utilizes a consolidated database (CDB) to optimize device client access and minimize back-end resource utilization.
- Device connection services – supports connections from various different platforms and operating systems with different communication styles.
  - Replication-based synchronization – A sychronization method where cached data is downloaded to and uploaded from client database to server via replication. Typically, mobile replication-based synchronization is used in occasionally connected scenarios.
  - Messaging-based synchronization – In flight messages are queued in a messaging cache. Synchronization occurs as messages are delivered to the device. Typically, mobile messaging-based synchronization is used in always available and occasionally disconnected scenarios.

See also

- *Server Administration Overview* on page 127
- *Data Mobility* on page 19

Application Development Environment

Unwired Platform incorporates the Eclipse-based Sybase Unwired WorkSpace application development environment.

Sybase Unwired WorkSpace is an extensive set of tools that simplifies and abstracts back-end system connections, and provides a uniform object view of all enterprise data and transaction objects.

The back-end development environment also includes device application development tools that provide rapid application development and device/emulator testing support for RIM® BlackBerry®, Microsoft® Windows Mobile, and Apple® iPhone® platforms.

Development is supported in a platform-neutral, as well as platform-specific, choice of languages and frameworks. For example, a Windows Mobile developer can use native C# and .NET Compact Framework programming. The same holds true for BlackBerry and iPhone platforms.

The development environment provides an open, flexible platform, which enables you to integrate third-party device application components and development tools, without being locked in to a particular set of development tools. For Windows Mobile application

development, you can install Sybase Unwired Platform Windows Mobile .NET Component in your Microsoft Visual Studio installation.

Sybase Control Center

Sybase Control Center (SCC) is the primary tool for administering and monitoring the Unwired Platform environment.

When it is installed as part of a Deployment Edition, Sybase Control Center can be used by administrators to perform full-spectrum support for Unwired Server-enabled mobility environments.

When it is installed as part of a Development Edition, Sybase Control Center can also be used by developers as part of the testing environment to deploy packages to a development-only Unwired Server. If the development environment is shared, the administrator uses Sybase Control Center to configure a common, shared Unwired Server.

Note: Configure a security provider for Sybase Control Center in a production installation of Unwired Platform. See *System Administration > Security Administration*.

See also

- *Cluster Administration Overview* on page 126
- *Server Administration Overview* on page 127
- *Domain Administration Overview* on page 138
- *Data Management Overview* on page 158
- *EIS Connection Management Overview* on page 146
- *MBO Package Management Overview* on page 150
- *Mobile Workflow Package Administration Overview* on page 153
- *Device and User Management Overview* on page 170
- *System Monitoring Overview* on page 185

Sybase Control Center Support for Unwired Server Administration

Sybase Control Center uses the Sybase Unified Agent framework, which provides a common set of server management services.

These common server management services allow you to:

- Perform a resource discovery to detect new Unwired Server instances on your network.
- Control Unwired Servers on your network from a single console.
- Deploy and manage replication, messaging, and workflow packages and their properties.
- Manage server connections used to connect Unwired Server to the enterprise data source.
- Create and manage device registrations for messaging and workflow packages.
- Create domains to partition Unwired Platform for multiple tenants, as well as create and assign domain administrators for that purpose.

- Monitor the system.

The framework consists of various components:

- A database to record and track plug-ins and managed resources.
- Sybase Unified Agent configured ports.

Sybase Control Center Functionality Not Applicable to Unwired Platform

Sybase Control Center is a standard management framework used by multiple products, including Sybase Unwired Platform. Certain standard functions that appear in the user interface cannot be used to administer Unwired Platform.

The following Sybase Control Center features can be disregarded in the context of Sybase Unwired Platform:

- Alerts
- Schedules
- Heat charts
- Historical performance monitoring
- Logging

These features either do not apply to Sybase Unwired Platform or are redundant due to custom functionality implemented in place of standard functions. The inapplicable Sybase Control Center functionality cannot be removed, as it may be required by other Sybase product servers also using Sybase Control Center.

Runtime Databases

Unwired Server uses multiple data stores to manage system and application data. These databases collectively make up the Unwired Platform data tier. In a production environment, the Unwired Server must be installed on a 64-bit host.

Depending on the type of application you are creating, there are different data stores for each:

- The consolidated database (CDB) acts as the Unwired Server cache and is used to store metadata and runtime data.
- The messaging database is used to store in-flight messages for messaging-based synchronization applications.

The remaining databases in the system are used to support the correct functioning of the Unwired Platform:

- The cluster database stores metadata on components deployed to a cluster.
- The monitoring database captures events for Unwired Platform components and deployed applications.

The Unwired Platform data tier can be installed separately from the server tier in a cluster. See *System Administration > Systems Design*.

Consolidated Database

The consolidated database (CDB) is a runtime cache database used by Unwired Server.

The CDB is a required component of Unwired Platform. By default, an embedded SQL Anywhere® database server is used as the CDB. However, during installation you can configure Unwired Server to use an existing SQL Anywhere instance as its CDB server.

Administration considerations

If you install multiple Unwired Servers in a load-balancing cluster, all Unwired Servers in the same cluster must share a CDB; however, in this scenario, a CDB failure can introduce a single point of failure for Unwired Platform. To mitigate this risk, you can run the CDB in failover mode using a shared-disk cluster. For information about implementing server and data tier clusters with optimal redundancy, see *System Administration > Systems Design*.

Depending on you environment, the location of the CDB database file varies:

- In cluster environment, the default file location is `<UnwiredPlatform_InstallDir>\Data\CDB\default.db`.
- For all other environments, the default file location is `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\data\default.db`.

If you choose an existing database instance as your CDB server, ensure that the database is configured to be case-insensitive; otherwise, issues with primary keys may arise during operation of Unwired Platform. In Unwired Platform, consolidated databases and client databases are configured as case insensitive by default. This configuration requires that primary keys in the datasource also be case insensitive. Otherwise, if there are two records with primary keys that just have case differences in the back-end system, they are treated as one record when loading them into the consolidated database. In this case, the second primary key overwrites the first that may exist in the consolidated database already.

Runtime data in the CDB

Besides acting as the synchronization cache for mobile business object (MBO) data, the CDB also contains metadata and runtime data, including:

- Unwired Server properties
- Push subscriptions and status
- Synchronization timestamps for each device client
- User data, such as personalization keys and device tracking information

See also

- *Database Clusters* on page 48

MBO Data in the CDB

A data cache is a copy of MBO data that is stored in a specific area of the consolidated database (CDB). It is used as the data repository for replication and messaging MBOs that are deployed

to Unwired Server. "CDB" and "cache" and "MBO data" can sometimes used interchangeably, even though the CDB includes runtime data as well.

When cache data is updated (either with an on-demand or scheduled cache refresh), the remote client database eventually retrieves the updated data from the server's copy of MBO data in the CDB by synchronization.

By giving applications a normalized and uniform view of corporate data, organizations can:

- Lower the barrier to data behind corporate firewalls
- Support development of mobile applications that interact with multiple enterprise back-ends
- Reduce back-end load caused by device client requests

Messaging Database

The messaging database stores in-flight messages in a messaging cache as part of a queue so they can be delivered.

By default, an embedded Advantage Database Server® is used as the messaging database.

Administration considerations

Depending on your network or system topology, install the messaging database either with the Unwired Server, or separately, on its own physical hardware. If you install multiple Unwired Servers in a load-balancing cluster, all Unwired Servers in the same cluster must share the messaging database. Sybase recommends that you install the messaging database separately on a failover cluster that uses a shared-disk array. You can install the messaging database and consolidated database together on the same physical hardware.

Depending on you environment, the location of the messaging database file varies:

- In cluster environment, the default file location is `<UnwiredPlatform_InstallDir>\Data\Messaging\`.
- For all other environments, the default file location is `<UnwiredPlatform_InstallDir>\Servers\MessagingServer\Data`.

Data in the messaging database

The messaging database include:

- Permanent store of device connection information – all registered messaging devices store their connection information in this database. During the first device connection, a physical device identifier is obtained by the messaging runtime client and is stored together with the connection information. Subsequently, the physical device identifier validates the device on every connection. Unwired Server accept messages from the device, but only when the identity is valid. This validation occurs even before the actual mobile application is authenticated by an Unwired Server security provider.

- Transient store for messages – the transient store in the messaging database holds business data only for a short amount of time. Once the data is delivered to the device, Unwired Server deletes the data from this database.

Cluster Database

The cluster database is, by default, a SQL Anywhere database used by the Unwired Server and associated command line utilities. It contains configuration information about the cluster for which the database is installed, as well as data used to coordinate cluster components.

A cluster database consists of a SQL Anywhere® database file and the server that serves the file.

Depending on you environment, the location of the cluster database file varies:

- In cluster environment, the default file location is `<UnwiredPlatform_InstallDir>\Data\CDB\clusterdb.db`.
- For all other environments, the default file location is `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\data\clusterdb.db`.

When a component starts, the utilities consult the cluster database. If there are cluster changes since the previous startup, the command line is modified accordingly. Various Unwired Servers then negotiate to determine which server is the primary server for the cluster.

Administration considerations

The `sup.properties` file is a copy of information from the cluster database that facilitates scripting. This file is located in *`<UnwiredPlatform_InstallDir>`*`\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer`.If you are troubleshooting the cluster, always check the cluster values in this database first.

Unwired Servers use the cluster database to negotiate which server acts as the primary server for the cluster. Sybase Control Center uses the primary server to coordinate cluster-wide administrative tasks such as package deployment, or connection configuration. Secondary Unwired Servers then monitor the cluster database to detect changes, which they then synchronize against to maintain consistency across the cluster. You can confirm the primary/secondary status in Sybase Control Center adjacent to the server display name in the left navigation pane of the Unwired Platform administration console.

Unwired Platform supports only SQL Anywhere cluster databases.

Data in the cluster database

The cluster database records configuration properties that are set for each component. The information is populated during installation, and then maintained over time as administrators use Sybase Control Center to change the production environment.

Specifically, this database includes:

- Configuration information for components in a cluster
- Installation details for components and the corresponding operational state of each

Monitoring Database

A single monitoring database stores data related to replication and messaging synchronization, queues status, users, data change notifications, and device notifications.

By default, the monitoring database uses the same database server instance as the consolidated database and cluster database.

Administration considerations

Administrators can install a monitoring database on host that is different from the consolidated or cluster database host. A monitoring database can be one of:

- A new SQL Anywhere database – a SQL Anywhere database used only by Unwired Platform monitoring. This is the default installation for Unwired Platform.
- An existing SQL Anywhere database – a SQL Anywhere database already used in your environment. You must set up Unwired Platform to use this database by configuring the database location.

  Note: The database server version must match the version used by this release of Unwired Platform. See the *Installation Guide* for details.
- An existing Adaptive Server® database – an Adaptive Server database already used in your environment. You must set up Unwired Platform to use this database by configuring the database location.

A monitoring database is shared by all server instances in a cluster. Consequently, administrators must allocate and manage the database depending on the monitoring configuration and system load.

If necessary, you can manually delete data, or you can use Sybase Control Center to allow Unwired Server to automatically purge the data. You can schedule a periodic and automatic flush of monitoring data that is captured in memory and move the data to the underlying monitoring database. This allows for capturing monitoring data without performance degradation of the client applications. If there are high levels of activity, Sybase recommends that you install the monitoring database on high performance hardware, and that you configure optimal flush settings for this environment. See *System Administration > Systems Maintenance and Monitoring > Monitoring Unwired Platform*.

See also

- *Monitoring Database Schema* on page 339
- *Setting Up an Existing Database for Monitoring* on page 78
- *Configuring Monitoring Performance Properties* on page 190

Device Clients

A device client is the mobile device a user employs to access your enterprise resources remotely. When a device becomes a client to the Unwired Platform, the device is given an ID, which is typically distinct from the device users identity.

Unwired Platform supports all major device platforms at various levels with respects to development paradigms and communication styles. Device clients must be provisioned with the client infrastructure, which typically includes:

Client software	How its installed
Afaria client	Download and install this client to manage and provision devices with applications over-the-air (OTA). If you are using Afaria, this is required.
Messaging or mobile workflow runtimes	Install the corresponding runtime (that is, use mobile applications that are using messaging-based synchronization, or mobile workflow forms on the device).
Messaging device application or mobile workflow forms	Developers create the native device application (messaging-based or replication-based synchronization)or mobile workflow form. Administrators provision and install device applications once the Afaria client is installed on the device.
Replication device application	Developers create the device application. Like the messaging application, administrators provision and install device applications once the Afaria client is installed on the device.

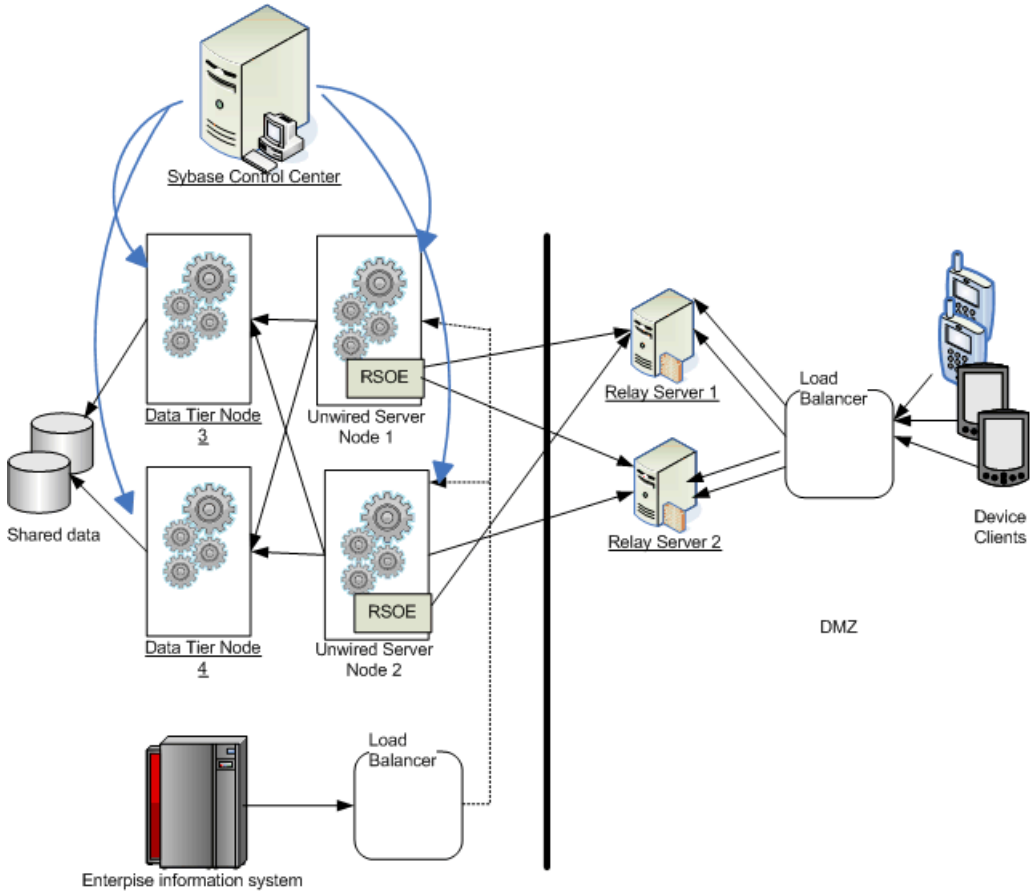
See also

- *Device and User Management* on page 169
- *Device Provisioning* on page 176
- *Mobile Workflow Package Administration Overview* on page 153
- *MBO Package Management Overview* on page 150

Relay Server

The relay server, a required component in a highly-available production environment, adds an extra layer of security and load balancing to the server environment. This server component, which is deployed into the enterprise DMZ, allows secure communications between devices and Unwired Server components across the firewall.

The relay server enables an outbound connection model. The Unwired Platform components make outbound connections into the enterprise DMZ using the secure HTTPS protocol. When using this component, connections from all device clients end within the DMZ of your enterprise, as shown by this architectural diagram:



The relay server also supports load balancing in your Unwired Platform installation by accepting requests from remote clients, and then forwarding the requests to a farm of Unwired Servers. If you are using Afaria, you can configure your Afaria environment to share one relay server.

Sybase offers two relay server options:

- For development and testing – the Sybase-hosted relay service, an Internet-hosted relay server that supports an Unwired Server cluster with a two-server limitation.
- For production – the Sybase-installed relay server, which supports Unwired Server with an unlimited number of redundant servers.

The relay server is implemented as a Web extension or a plug-in that runs in a Web server. Unwired Platform supports two Web servers: IIS on Windows, and Apache on Linux. The relay server accepts device client requests and distributes them across a farm of Unwired Servers and Afaria servers. Each individual Unwired Server, Unwired Server embedded component, and Afaria server must run a relay server outbound enabler (RSOE), which

establishes a permanent connection to the relay server on behalf of the server or embedded server component.

See also

- *Relay Server Setup* on page 62
- *Relay Server Documentation* on page 63
- *Relay Server Configuration Files* on page 336
- *Relay Server Utilities* on page 297
- *Configuring Relay Server Outbound Enabler Logging* on page 232

Afaria

Afaria is an optional component that is separately licensed and only supported on 32-bit hosts. Use the Afaria with Unwired Server to perform advanced mobile device management and provisioning in your production environment.

In an Unwired Platform Environment, administrators often use Afaria for:

- Unwired Platform devices provisioning. See *System Administration > Systems Administration > Users and Device Adminsitration > Device Provisioning*.
- Extended device security. See *System Administration > Security Administration > Security Layers > Data Security Setup*.

However, because of platform requirements for Afaria and Unwired Platform, Afaria must be installed on seperate hosts.

See also

- *Afaria Setup* on page 82
- *Afaria Documentation* on page 83
- *Afaria Provisioning and Mobile Device Management* on page 177
- *Afaria Licenses* on page 243

Afaria Administrator

Afaria Administrator is the tool for advanced management and provisioning capabilities for devices that are also clients of Unwired Server. Use Afaria Administrator to extend your administration scope beyond Unwired Platform environment adminstration and MBO package deployment.

Afaria may not always be required in smaller deployments of Unwired Platform. However, depending on the scale and scope of your license, you may want to use Afaria with Unwired Platform to simplify the management of many devices and device users.

CHAPTER 3 Enterprise Mobility

Extending your enterprise business processes, data, and information away from the office entails many technical challenges.

- Device heterogeneity
- Data partitioning for mobile applications
- Data transport and application security
- Bandwidth, latency, and connectivity issues
- Managing devices, and the application life cycle

Each challenge presents complexity to the IT organization. These complexities may include, but not be limited to, for example, lack of integrated solutions, heavy upfront costs for large-scale customization requirements, and the lack of repeatable development and deployment models.

Sybase Unwired Platform meets those challenges by hiding system complexity while efficiently and flexibly handling complex mobility issues. Unwired Platform provides many opportunities for productivity gains through a well-defined life cycle, repeatable use of existing investments, and mobility knowledge.

Data Mobility

Businesses increasingly need to connect people and data inside *and* outside the bricks and morter of their corporate offices.

Seamless connectivity and access to real-time information creates the ability to take action immediately. Whether the goal is an up-to-date salesforce or a health care work receiving immediate updates on a client, each of these organizations constitutes an enterprise geared toward a need to make their information mobile and timely.

Data synchronization is fundamental to mobilizing enterprise data in Unwired Platform. Just-in-time information and seamless connectivity must be tailored to these enterprises via the mechanisms and features of data synchronization. Connecting your mobile workforce to existing but disparate enterprise information systems and diverse applications inside and outside of the four walls is no simple feat. It is necessary to ensure that they gain access to the right information, at the right time, to make the right business decisions at the point of business activity. Administrators play a key role in ensuring this need is realized by configuring all aspects of the mobility environment, so that the system is well integrated, managed, and secured.

See also

- *Data Management Overview* on page 158
- *Database Clusters* on page 48
- *Backup and Recovery* on page 234

Administration and Development Coordination

Developers play an important role in the design of the mobile application. But administrators control other important behaviors after a package is deployed. Understanding the dynamic between these roles is critical in a mobile environment, especially with respects to data updates.

The distinction of these roles can be summarized as follows:

- Administrators need to co-ordinate the behavior of how information is relayed to and from the device user. Issues of synchronous versus asynchronous synchronization schedules, of state data in the consolidated database, and of outbound device client notification are the decisions of the administrator after packages are deployed. Because these issues affect the design and user-interface of the client application, the terms of this setup can be negotiated and agreed upon with development stakeholders.
- Developers need to design mobile business objects so they get data to and from the consolidated database and the enterprise information systems (EIS) effectively. However, they still need to understand where the client-side synchronization ends up. For example:
  - Is data going just to the consolidated database or is it written to EIS synchronously?
  - How stale data could be is a primary concern that must be addressed by the developer (that is, will data be updated regularly or predictably by a synchronization interval or more immediately by a data change notification)?

Data Mobility Patterns

Administrators of an Unwired Platform environment have the unique goal of configuring either a development environment or a production environment, so that it supports the mobility pattern being used by the organization.

Unwired Platform currently can use any combination of the following:

- Data Virtualization – sometimes referred to as Information-as-a-Service or Data-as-a-Service. Data virtualization eases the impediments to enterprise-level data integration by decoupling data from EIS servers or applications and storing it in the Unwired Server cache in the consolidated database (CDB) on the middleware layer of the network architecture. This allows device client users to have "a single source of truth", by allowing multiple applications to interact with the same virtualized layer, which helps to ensure data consistency.

  For this pattern, system management goals include:

  - Normalizing multiple back-end EIS data sources as relational data in the Unwired Server CDB cache.
  - Configuring and injecting security provider services device to control client access to those data sources.
- Data Publication – is the pattern that is used to define how the EIS data that is made available to device clients. To allow device applications to synchronize a slice or subset of data, administrators can create a synchronization group. A synchronization group identifies data that is to be synchronized.

  For this pattern, system management goals include:
  - Making enterprise data available in the most appropriate form for device applications.
- Data Subscription – is the mechanism that links the client with a publication and a registered Unwired Platform user that allows the data described by the publication to be synchronized.

  For this pattern, system management goals include:
  - Planning the subscriptions design to create subset of enterprise data.
  - Creating notifications for high-value data changes to increase data currency on the device.
- State Replication – In Unwired Platform, only the metadata table is replicated directly from device database to the Unwired Server CDB.

  For this pattern, system management goals include:
  - Maintaining the integrity of Create-Update-Delete operations by supporting the development approach used. There are two: either set up data as read only and create an application that never changes data programmatically on the device (thereby allowing data only to be updated by a synchronization download); or, allows only user-specific remote data to be modified (for example, a customer's contact information update should be visible immediately on device even though an operation replay to the enterprise is still pending).
- Operation Replay – Used with the data virtualization pattern, operation replay pattern support both service oriented architectures and ad hoc application integration techniques. In this approach, the mobile application executes an operation on the client side; the mobile infrastructure relays the operation back to the enterprise, and executes the server portion on behalf of the mobile application.

  For this pattern, system management goals include:
  - Ensuring that developers are able to propagate actions to enterprise for execution in the MBO by maintaining a pool of datasource connections.
  - Supporting conflict resolution during synchronization uploads.

Network Availability Models

Developers and administrators must consider availability of a network connection to device application users when designing and developing mobile applications.

The Unwired Server Platform provides data access and reliability through a number of embedded technologies and features regardless of network availability, including:

- Globally Unique Identifier (GUID) – client-side generated ID that does not change when persisted in either the local device database or the Unwired Server cache (CDB). This insures the identities are correct when saved, or refreshed. And the GUID can be referenced when reconnecting to Unwired Server.
- Publish/subscribe model – messaging mobile applications use the publish/subscribe model, by which Unwired Server keeps track and pushes only the content to which the client subscribes since the last connection.
- Persistent storage, caching, and notifications – managing multiple client synchronizations, maintaining data integrity, and notifying clients of data changes when they reconnect is managed by Unwired Server and the cache (CDB). The device's database allows off-line work to continue until updates can be made.
- Transactional integrity – MBOs are replicated and transactional integrity is maintained by the database used as the CDB.
- Conflict detection and resolution – a number of underlying technologies detects and resolves data conflict.

Always Available

In an always available environment, mobile application clients almost always have an available wireless connection to Unwired Server.

Message-based mobile applications are ideal solutions in an always available environment, in terms of both cost to develop, and that they can take advantage of a variety of data sources compared to a strictly replication-based mobile application.

See also

- *Message-Based Synchronization* on page 24
- *Messaging Devices* on page 171
- *Subscriptions* on page 174

Occasionally Disconnected

In an occasionally disconnected environment, mobile application clients, usually have an available wireless connection to Unwired Server.

Connected device application users can do most of their work online, synchronizing data at will. These users usually require locally stored business data.

Create mobile applications that work in either synchronous or asynchronous mode. From the device perspective:

- Asynchronous – the application does not wait for Unwired Server to report on the success or failure of an operation. Asynchronous is useful when the application is in a disconnected state.
- Synchronous – the application waits until Unwired Server performs the action, at which point a new screen can optionally open, depending on the results.

See also

- *Replication-Based Synchronization* on page 23
- *MBO Package Management Overview* on page 150
- *Subscriptions* on page 174
- *Replication Devices* on page 173

Occasionally Connected

In an Occasionally connected environment, mobile application clients, usually do not have an available wireless connection to Unwired Server.

Disconnected device application users do most of their work offline, then synchronize when they are connected to the network. These users require locally stored business data, which enables them to perform their job anywhere.

Synchronization Methods

Developers can use either replication-based or message-based synchronization to move data and transactions between device application clients and Unwired Server.

The choice depends on the target device platform, application requirements, target platform, and the nature of data changes and activity between Unwired Server and clients, for example, mobile workflow forms always use message-based synchronization.

Unwired Server manages and maintains data freshness between multiple data sources and device application through synchronization.

Replication-Based Synchronization

The replication-based synchronization model uses relational database replication, and is session-based to reduce the overhead required to maintain a continual connection. During the session, the clients submit pending operations and obtain the latest changes.

The Unwired Server can send out-of-bound device notifications to the subscribing clients; that client can then initiate a request to get the changed data.

Typically, replication-based synchronization is used in an occasionally connected or occasionally disconnected environment. Replication-based synchronization works well in

situations where a large amount of data is stored on the device, continuous access to that data is required, and the data changes in bursts that require periodic synchronization between the client and server. This paradigm is supported on Windows, Windows Mobile, and BlackBerry device platforms

See also

- *Occasionally Disconnected* on page 22
- *MBO Package Management Overview* on page 150
- *Subscriptions* on page 174
- *Replication Devices* on page 173

Message-Based Synchronization

The message-based synchronization model is inherently asynchronous. The interaction, however, can either be synchronous, or asynchronous for messaging-based synchronization. During transmission, the client and server submit pending operations and obtain the latest changes.

Inherently asynchronous, data changes from the server to client are automatically sent and applied to client's local data store, and, similarly, operation executions from the client to server are sent in the background without requiring explicit action from device application. The developer can choose to keep transactions from being uploaded to the server by manipulating state of the submission (as pending).

Message-based services support a mobile application class that consumes message events for data and notifications or actions, and generates events to propagate changes in an always-on network availability scenario.

Message-based synchronization works well in situations that require functionality to send updates to the client intermittently with small data payload. As with replication-based synchronization, message-based synchronization can also store large amounts of data on the device, and data exchange may be more transient, mobile workflow-related, and occurs continuously throughout the life cycle of the application. This paradigm is supported on Windows, Windows Mobile, and iPhone device platforms.

See also

- *Always Available* on page 22
- *Messaging Devices* on page 171
- *Subscriptions* on page 174

Mobile Workflow Data Flow

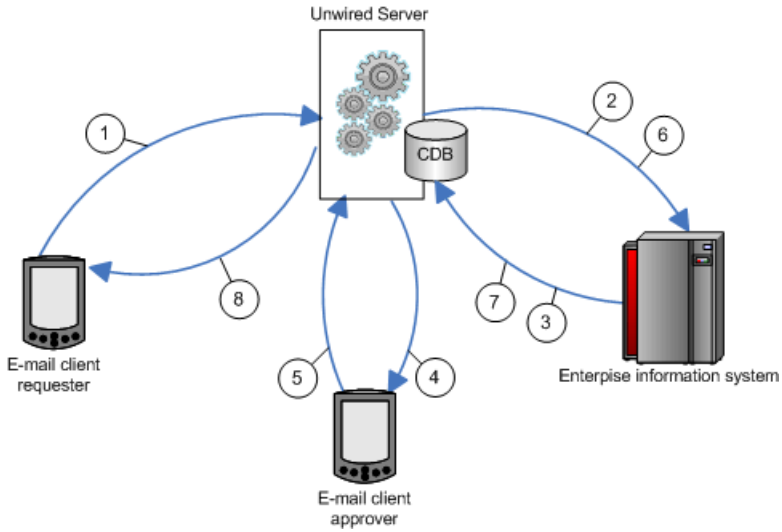
In the business workflow model, when you invoke an MBO operation located on Unwired Server using a Submit action, you can specify synchronous behavior (the messaging application waits for a successful or failed response from Unwired Server before proceeding). From an MBO/EIS perspective, Unwired Server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

Some of the differences between messaging mobile applications and replicated mobile applications include:

- The MBOs included in the application are no different than any other MBO: same types of parameters, attributes, CDB caching, synchronization methods, and so on.
- There is no permanent storage of the message. For example, a workflow application consists of the MBO portion, which is managed by Unwired Server, and the message portion. The message portion of the application is the transient store and forward system to deliver the messages reliably between server and device client, and takes advantage of the capability to build messages on the fly and send to the interested devices with them having to explicilty know or request it.

This workflow example is a travel approval application that includes:

- A TravelRequest MBO that includes:
  - dates, location, estimated costs, purpose, and a unique ID.
  - status and comment – included in the MBO definition but implemented by the business process widget.
  - An object query that returns a row based on the submitted ID.
- Two triggers:
  1. Sends an message to the approver when a new row is inserted into the MBO table.
  2. Notifies the requester when the status of the request has been updated by the approver.
- A business process widget that implements the status and comment portion of the application.



1. An e-mail requesting travel is submitted.

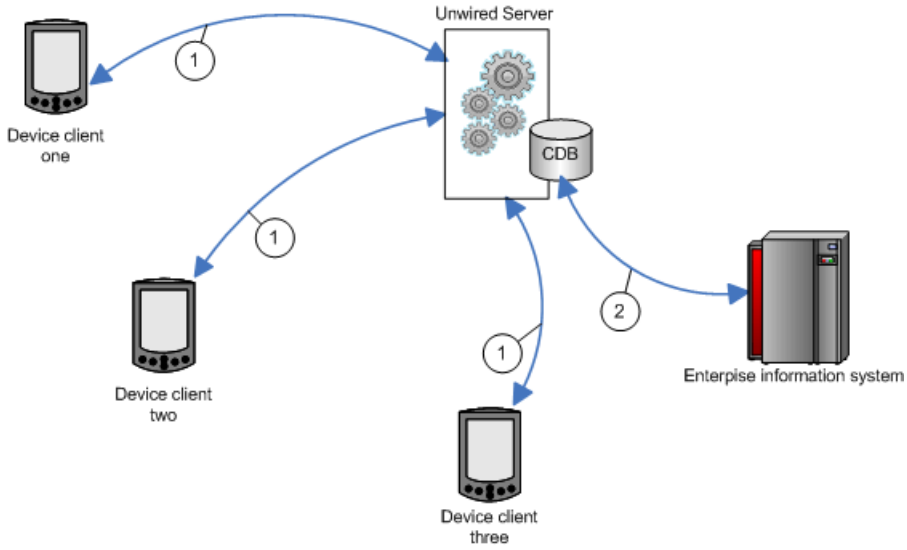
2. Depending on the data refresh schedule or the operation's cache policy, the cache in CDB will be updated.
3. Triggers a message to the approver.
4. The travel request is approved through e-mail.
5. The EIS is updated with the approved information.
6. Depending on the data refresh schedule or the operation's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

Data Synchronization and Data Refresh

Since dataset variations occur between multiple clients and the enterprise information system (EIS) data to which mobile business object (MBO) data is bound, synchronization is required to reconcile differences and bring each client into coherence with the working copy of the EIS data maintained in the consolidated database (CDB), before writing updates back to the EIS.

These terms describe maintaining data consistency:

- Synchronization – synchronize between the CDB and mobile-device applications. Synchronization transactions require a connection. If a mobile device does not have a connection to Unwired Server, synchronization cannot occur until a connection is established. However, data updates are aggregated and synchronized when a connection becomes available.
- Data refresh – also called cache refresh, synchronize between the CDB and an EIS. Because information is held in the CDB, even if the EIS server fails, the device still has read access to the data in the CDB.



1. Each client maintains one instance of the data. Similarly, there is only one version of the dataset in the CDB, and only one version in the EIS system.
2. Since variations occur between the different clients and the EIS data, synchronization brings each client into coherence with the working copy of the EIS data that is maintained in the CDB.

Synchronization and Data Refresh Triggers

Initiate synchronization and data refresh using a combination of methods to effectively meet mobile application and system requirements.

See also

- *Synchronization and Data Refresh Data Flow* on page 29
- *Synchronization and Data Refresh Strategies* on page 37

Synchronization Triggers

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

Table 2. Synchronization methods and triggers

Method	Description

Push	For the push method, either the the MBO developer or the administrator configures synchronization timing (on-demand or scheduled). Typically a refresh schedule or data change notification (DCN) is paired with a subscription template for a given MBO with push synchronization enabled. When MBO data in the CDB changes:
	• Notifications are sent at a set interval. The default is one minute and ends when the client acknowledges it has received notification.
	• Unwired Server determines when individual clients need to be notified of changes and can override device application settings and synchronize the device application with the contents of the CDB.
	• If Unwired Server does not force a synchronization, device application logic determines how to respond to the push notification. The device application developer can:
	  • Register to receive push notifications from Unwired Server.
	  • Implement a push listener.
	  • Implement logic to react to push notifications. For example, if certain data changes the device application synchronizes with the CDB.
	Note: Unwired Server initiates notifications only for replicated-based synchronization, while messaging-based synchronization pushes data to the device without notification.

Method	Description
Pull	For the pull method, the MBO developer configures the amount of data that is to be synchronized (through a combination of settings such as Synchronization group, synchronization parameters, data filter, and so on), and the device application developer adds the screens and logic that allows a user to pass synchronization parameters and attributes to trigger and control synchronization, including: • Starting the device application – this can automatically trigger synchronization. • Adding a synchronization event button to the device application – allows the device application user to synchronize based on how the synchronization event is configured. For example, the MBO developer could include a synchronization parameter that filters data displayed by the device application, or supply client parameters (username and password) by which the device synchronizes. • Device application logic – the device application developer adds logic that triggers synchronization based on an event.

Data Refresh Triggers

Define data refresh through mobile business object (MBO) settings, programmatically through the data change notification (DCN) interface, or through Unwired Server settings.

Table 3. Data refresh methods and triggers

Method	Description
DCN (development) | Push Listener (administration)	The enterprise information system (EIS) developer implements DCN using HTTP or HTTPS GET or POST methods. Depending on which is implemented, the administrator needs to configure the push listener synchronization gateway for the correct encrypted or unencrypted protocol chosen. The DCN can be initiated by a database trigger, stored procedure, or some other event to: • Notify Unwired Server that a particular MBO in the CDB needs to be refreshed. • Allow the EIS to invoke a particular MBO operation with a set of specified parameters.
Cache group	The MBO developer defines any number of cache groups to which one or more MBOs are added based on data refresh requirements. An update policy applies to all MBOs within a cache group.
Cache update policies	The MBO developer can add a cache update policy to create, update, or delete operations to control how any EIS affecting operation is applied to the CDB.

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

Synchronization and Data Refresh Data Flow

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

Sybase Unwired Platform supports both replicated and nonreplicated data within device applications. Except as noted, all references in the synchronization and data refresh overview refer to replicated data flow.

See also

- *Synchronization and Data Refresh Triggers* on page 27
- *Synchronization and Data Refresh Strategies* on page 37

Synchronization Data Flow

Synchronization is when a device application's data is updated with the contents of the Unwired Server cache (which is also called the consolidated database, or CDB).

Based on various cache settings, the enterprise information system (EIS) can update or refresh the cache during synchronization. Device application initiated synchronization occurs at the request of the user, through a menu button or triggered programmatically as the result of some application action or timer.

Filtering and Synchronizing Data

Use filters and paramters to synchronize selected subsets of data.

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

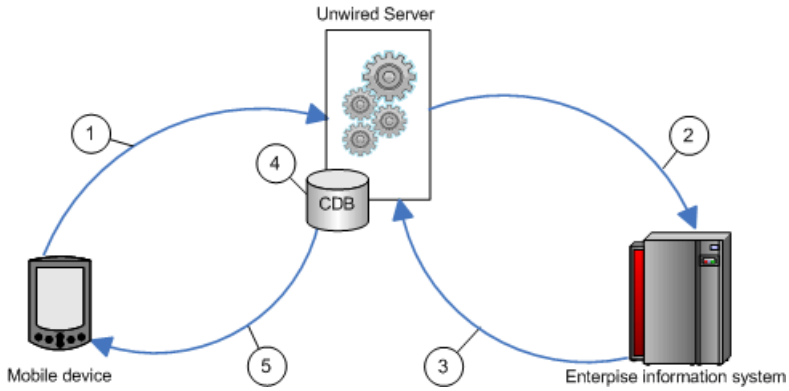
Load Parameter Data Flow

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

Pairing a load parameter with a synchronization parameter during mobile business object (MBO) development, indicates that the user will supply values for this parameter over time

and the aggregate set of data based on the values provided over time are synchronized with that device. If not paired (or mapped) to a synchronization parameter, no such synchronization filtering occurs for the device and the parameter is simply used to update the consolidated database (CDB) by retrieving a subset of data from the enterprise information system (EIS).

An initial read operation populates a CDB table with all rows of MBO data, which can be included in the data returned in synchronization requests made from one or more clients. In some cases, a load parameter is desired to refine the data requested from the EIS. Mapping the load parameter to a synchronization parameter partitions data in the CDB according to values sent from each device client.

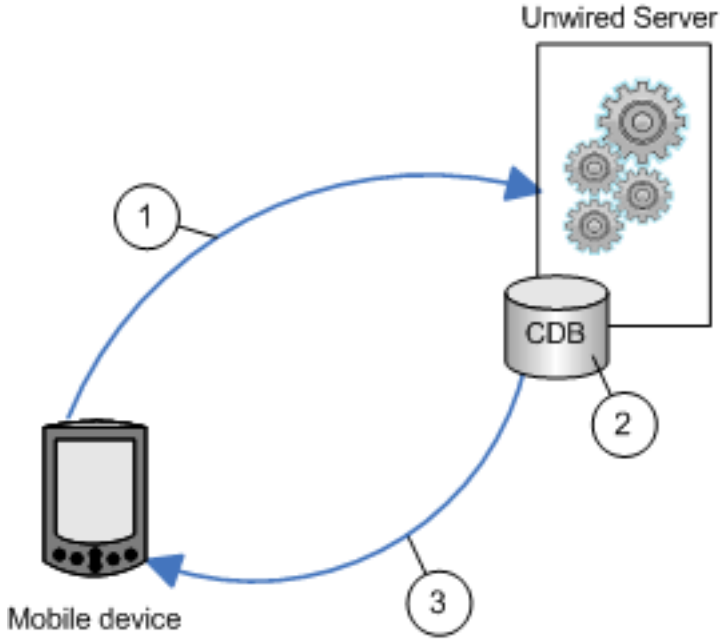


1. The user initiates a synchronization request and includes a parameter value, for example, a user name. Be aware that passwords should not be used as parameters.
2. If personalization keys are used as the load parameter, Unwired Server passes the query to the EIS. If the parameters are user credentials, they are validated by the EIS.
3. The EIS refreshes Unwired Server based on the parameter, for example, it refreshes data only for the validated user. If the parameter is region, and the parameter value is "western," only results for the western region refresh.
4. Unwired Server creates a partition (branch) with the results in the CDB for the validated user, or updates the partition if the user has previously synchronized.
5. Unwired Server synchronizes the device with the data in the CDB partition for that user.

Synchronization Parameter Data Flow

An attribute corresponds to a column in a table. A synchronization parameter maps to an attribute that acts as a filter or variable that lets you limit the data that is returned to the device to rows in the table based on a supplied value.

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



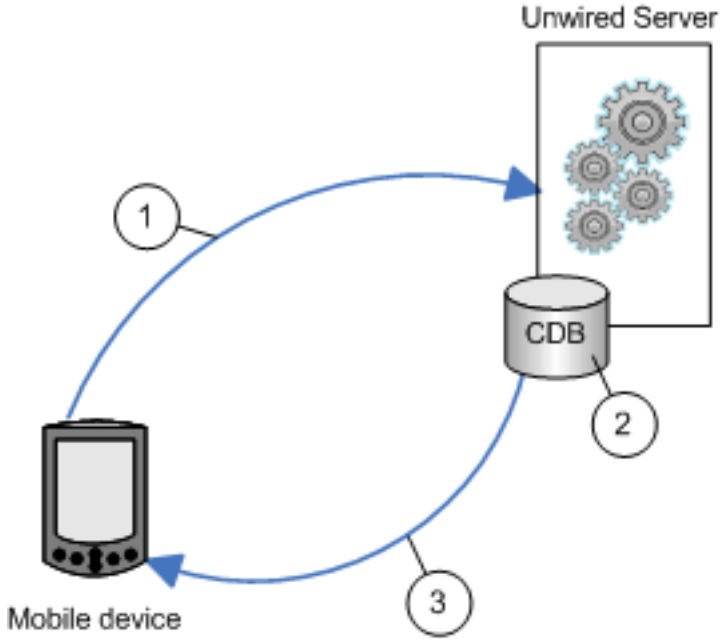
1. The user initiates a synchronization request that includes an attribute value (synchronization parameter).
2. Unwired Server filters the data in the CDB. For example, if the attribute is "country" and the user supplied the value "USA," only rows that contain "USA" are returned to the device.

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

Result Set Filter Data Flow

A result set filter is a custom Java class deployed to Unwired Server that manipulates rows and columns of data before synchronization.

Result set filters are more versatile (and more complicated to implement) than an attribute filter implemented through a synchronization parameter, since you must write code that implements the filter, instead of simply mapping a parameter to a column to use as the filter. See *Developers Reference: Server API*.

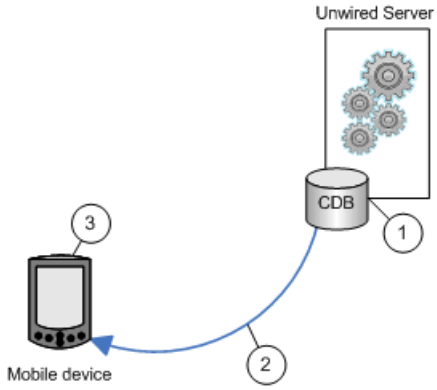


1. The device application initiates a synchronization request to a mobile business object that implements a result set filter.
2. Unwired Server applies the result set filter to the CDB data requested by the device application. For example, the result set filter combines two columns into one.
3. Unwired Server synchronizes the device with the results.

Synchronization Initiated by Unwired Server

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

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



1. Unwired Server detects a change in the data cache; for example, through a data change notification (DCN) or a data refresh.
2. Unwired Server notifies registered devices of changes to cached MBO data. If it is configured to do so, Unwired Server may force a synchronization with the device; for example, if the data is critical.
3. Implement logic in device applications to appropriately react to push notifications, if the Unwired Server does not force a synchronization.

Data Refresh Data Flow

Data refresh occurs when enterprise information system (EIS) data updates are propagated to the Unwired Server cache (also called the consolidated database, or CDB).

There are two general categories by which data refresh occurs:

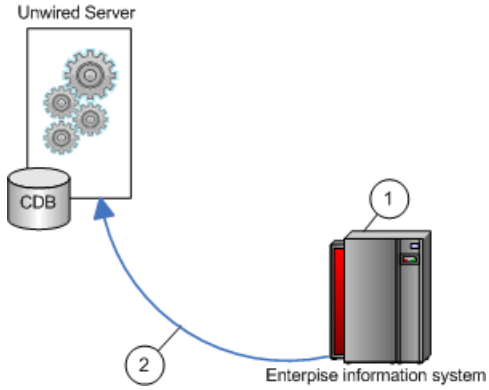
- Unwired Server pulls – pulls updates from the EIS either through Unwired Server configuration or policies defined by the MBO developer.
- EIS pushes – a DCN option that includes all information required for an update are pushed to Unwired Server.

Data Change Notification Data Flow

Data change notifications (DCNs) refresh data when a change to the enterprise information system (EIS) occurs.

DCN requests are sent to Unwired Server as HTTP GET or POST operations. Each DCN can instruct Unwired Server to modify cached MBO data.

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



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

Data Change Notification Interface

Data change notification (DCN) provides an HTTP interface by which enterprise information system (EIS) changes can immediately be propagated to Unwired Server.

All DCN commands support both GET and POST methods. The EIS developer creates and sends a DCN to Unwired Server through HTTP GET or POST operations. The portion of the DCN command parameters that come after `http://host:8000/dcn/DCNServlet` can all be in POST, or any *var=name* can be in either the URL (GET) or in the POST. The **authenticate.password** parameter is an especially good candidate for including in the POST method, as well as any sensitive data provided for attributes and parameters because the HTTP POST method is more secure than HTTP GET methods. See *Developers Reference: Server API*.

Note: Enter the HTTP request on a single line.

You must be familiar with the EIS data source from which the DCN is issued. DCNs can be created and sent based on:

- Database triggers
- EIS system events
- External integration processes

You can use DCN with or without payload to instruct Unwired Server to refresh data:

- DCN without payload – can only call MBO operations, and the name used in the DCN request must match that of the MBO definition:
  1. The DCN requester (which may or may not be the EIS for DCN) sends an MBO operation execution request, along with operation parameters, to Unwired Server.

2. Unwired Server executes the operation, (effectively calling the EIS operation), and updates the consolidated database (CDB), if needed, depending on the operation's Cache Update Policy.
3. Unwired Server returns a DCN status message to the requester.

Note: Be careful when naming MBO operations in Unwired WorkSpace. If "delete" is the name of the EIS effecting MBO operation, it can easily be confused with the direct cache-effecting operation named **:delete**.

- DCN with payload – calls only the two direct cache-effecting operations (**:upsert** or **:delete**), which always exist for an MBO, and are not related to user defined MBO operations.
  - :upsert – the message must contain name/value pairs for every required attribute, and the name must match the MBO attribute name exactly.
  - :delete – only the name/value pairs for the primary key column(s) need to be provided.

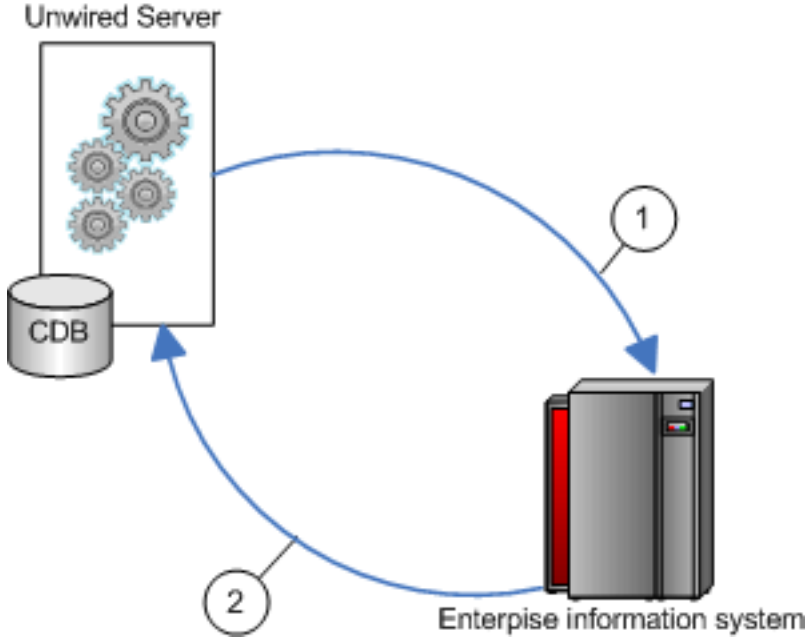
  These operations insert/update a row in the CDB or delete a row from the CDB. Calling either of these operations does not trigger any other refresh action:
  1. Some event initiates the DCN request (a database trigger for example).
  2. The Unwired Server cache is updated directly from the EIS. The actual data (payload) is applied to the cache, through either an **:upsert** (update or insert) or a **:delete** operation.
  3. Unwired Server returns a DCN status message to the requester.

See also

- *Data Change Notifications* on page 163

Cache Group Data Flow

A cache group policy determines the frequency and the level to which mobile business objects (MBOs) belonging to that group are refreshed.

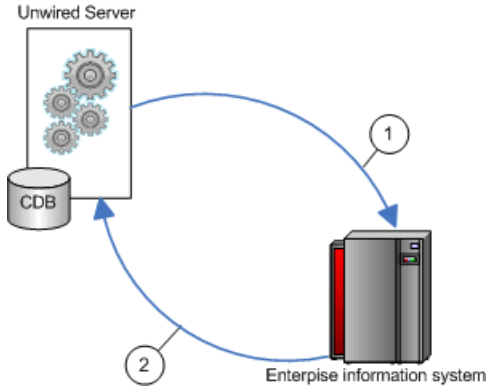


1. The deployed MBO triggers a cache update depending on the cache group to which it belongs.
2. The CDB is updated based on the cache group settings.

Data Refresh Initiated by Unwired Server

Configure Unwired Server to "poll" the enterprise information system (EIS) at scheduled intervals to determine if data has changed. If it has, the EIS refreshes cached MBO data.

The Unwired Server administrator uses the Administration Console to schedule data refresh intervals for a given MBO. This simple and flexible data refresh strategy uses more system resources than data change notification (DCN).



1. Unwired Server polls the EIS at an interval determined by the Unwired Server administrator.
2. If data changes, the CDB is refreshed.

Synchronization and Data Refresh Strategies

Combine synchronization and data refresh techniques and strategies to successfully meet the business needs of the mobile application while effectively utilizing resources.

- Design and planning – carefully consider:
  - Limiting data in the mobile business object (MBO) to what is required to meet business needs.
  - The size and scope of the mobile application.
- Timing – coordinating device application synchronization with data refresh to achieve optimum results.
- Methods – use a combination of methods to control how much data in the Unwired Server cache is updated when EIS data changes. It is relatively simple to design your system to invalidate MBO data in the CDB and refresh it from the EIS whenever EIS data changes, but this can be inefficient.

See also

- *Synchronization and Data Refresh Triggers* on page 27
- *Synchronization and Data Refresh Data Flow* on page 29

Synchronization Scenarios and Strategies

Mobile application development and administration settings allow you to decide when and how to synchronize and refresh data.

The following table describes strategies to consider when designing and developing your mobile applications. Since most mobile applications include several (if not many) MBOs, you will combine various strategies.

Table 4. Synchronization and data refresh strategies and examples

Scenario	Strategy
Data changes irregularly in the enterprise information system (EIS), and data is noncritical.	Either: • Unwired Server initiates a data refresh once a day at off-peak hours that invalidates and refreshes all MBO data, or • The MBO belongs to a synchronization group that includes a daily interval. For example, an organization bulk-loads data by performing a data refresh at the end of the work day. Field service personnel filter an attribute and synchronize (synchronization parameter=region) at the beginning of their day to synchronize only data of interest.
Clients must immediately synchronize critical EIS data changes.	You could either implement a cache update policy that immediately applies the results of the MBO operation to the CDB. Or, Implement a data change notification (DCN) that performs a data refresh of targeted MBO data. Unwired Server then sends notifications to registered device clients, and optionally force a synchronization to targeted devices. For example, administrators, professors, and others on a college campus have registered their mobile devices with campus police. When an emergency call from campus is received and entered into the system, a DCN updates the cache. Unwired Server forces synchronization with registered devices.
EIS data changes frequently.	Either: • Implement an Unwired Server scheduled refresh that polls the EIS at specified intervals and updates the CDB when changes occur, or • Define various cache groups for MBOs that refresh data as necessary. For example, set a short data refresh interval to receive up-to-date quotes for an equity tracking mobile application (stocks, bonds, and so on), which also allows users to buy and sell equities.
Device application users change EIS data frequently.	When defining MBO create, update, and delete operations, include a cache update policy for EIS modifying operations. Choose a policy that updates the CDB only with necessary changes. For example, a mobile application contains regional sales information. When you make a sale to a new customer, the MBO inserts a new row in the corresponding EIS table. To see changes related to your customers only (rows that contain your territory ID), use the apply operations parameter policy when defining the MBO.

Scenario	Strategy
A mobile application supports thousands of individual users, each of which has a set of user specific data.	The MBO developer creates personalization keys that are used as client parameters (user_name and password), which are validated by the EIS. For example, a mobile application used for retail sales maintains login information for validated customers that provides access to account information, shopping cart, wish list, and so on.
A mobile application has both static and changeable data.	The MBO developer configures two cache groups, designed to refresh Unwired Server cache (CDB) for each MBO based on the frequency of EIS data changes to which each MBO is bound. For example, a mobile application contains a sales_order MBO with a many-to-one relationship to the product MBO. While sales_order related data changes often, as sales are made, product data does not. The MBO developer establishes two cache groups: **1.** The sales_order MBO data cache updates hourly. **2.** The product MBO data cache updates daily.
A Human Resources department wants to implement a mobile application used to request and approve travel and expenses.	The message-based mobile application uses both replicated data (managed within the MBO) and nonreplicated messaging data. Unwired Server pushes changes/updates to device application users. For example, the application includes MBO bound data (calendar with requested dates, total cost, and so on). The messaging portion includes additional information including the message. Once requested and e-mailed to the manager, the recipient (manager) approves dates and expenses. The MBOs are updated in the EIS (replicated and synchronized), while the message is not.

Bulk Loading Cache Strategy

Bulk loads transfer large quantities of data into the consolidated database (CDB), which acts as the Unwired Server cache. Because of the demands this places on resources, special consideration needs to be given when using this strategy.

When Used

Bulk loads are typically used when:

- Creating an initial data set of reference data for the device application. Bulk loads are ideal for data that does not change often.
- The structure of EIS data changes and the application data structure needs to be normalized on the device.
- Repairing a complete data set when corrupted on the device or on the CDB.

How Used

For each MBO, the developer selects the largest set of data that is applicable to most mobile users and designs the MBO to load data into the Unwired Server cache all at once. The

developer then needs to define synchronization criteria for the MBO, so that synchronization parameters index into the cache to retrieve device-relevant data subsets. Good for data where the device is primarily driving change (work orders) or reference data that does not change often

Considerations

- Preloads the cache up front and isolates specific device invocations from the enterprise application.
- Large up-front cost of data that may never be used. Bulk loads are best performed when minimal amount of users are connected to the EIS server.

On-Demand Loading Synchronization Strategy

On-demand loads transfer a specific subset of data when the exact data required cannot be predetermined at design time, especially when space and performance demands imposed by bulk loading is an issue in the production environment

When Used

- Data is highly customizable. Very few people share the same information needs.
- Data changes frequently.

How Used

For each MBO, developers define parameters based on context variables on the device (for example, personalization variables). The MBO then directly passes the context variables to Unwired Server to build up the cache over time, based on user properties. Good where the exact data to be referenced cannot be predetermined and where space is an issue

Considerations

- Loads the cache over time and eliminates the need to load vast quantities of data that may never be referenced.
- More invocations to the enterprise application; may need cache seeding for newly provisioned devices

The Impact of Synchronization and Data Refresh

When designing and developing your mobile application solutions, consider the impact of varying synchronization and data refresh methods.

Table 5. Impact of various data refresh methods

Data refresh method	Implication
Data change notification (DCN)	Requires a developer familiar with the EIS from which the DCN is sent. Provides more flexibility than a scheduled refresh, but is more complicated to implement. HTTP GET methods are less secure than HTTP POST.
Unwired Server scheduled data refresh	Easily implemented by the Unwired Server administrator. Less targeted than DCN or a cache group. Uses more system resources since it must periodically query the EIS for changes.
Cache group	Easily implemented by the MBO developer. A cache group is a collection of cache tables to which a common refresh policy is applied.
Cache update policy	Easily implemented by the MBO developer. Updates the CDB for an EIS data effecting operation (create, update, or delete) based on the policy associated with the operation. For example, you could update a modified row or invalidate and refresh the entire MBO.
Load parameter	Easily implemented by the MBO developer. Load parameters map to data source arguments and refresh data based on the supplied value. They can be used alone or mapped to synchronization parameter to control both data refresh and synchronization.

Table 6. Impact of various synchronization methods

Synchronization method	Implication
Initiated by Unwired Server	Easily implemented by the Unwired Server administrator. The primary consideration is balancing performance with resource usage when determining how frequently synchronization is initiated, and to how many registered devices: • Download from Server – control when data is downloaded to remote devices. For example, if set to 10 minutes, the server notifies the client of data updates at most every 10 minutes, even if data changes more often. • Device notifications – notifications continue at a predetermined interval until acknowledged by the registered device, even when devices are disconnected or out of range.
Initiated by the device	Options include: • Filtering results – implemented by the MBO and device application developer. The MBO can be configured through load and synchronization parameters to have the CDB maintain partitions for each user, resulting in a growing list of clients who synchronize a filtered data set. • Custom listener – implemented to listen for messages sent by the notifier. When the listener receives a message, it can be programmed to initiate synchronization.
Defined within the MBO	Options include: • Synchronization group – groups MBOs by synchronization needs. • Synchronization tab – defines synchronization requirements for individual MBOs.

CHAPTER 4 Systems Design

Different Unwired Platform deployment architectures support the different stages of the development life cycle, by mapping the logical architecture of Unwired Platform components to a physical computing environment, according to the requirements specified for each.

Most customers needs are met by using one of these environment types: single node environments (sometimes known as trial environments), shared development and test environments, simple redundancy environment, full-scale (or optimal) redundancy environments, and even multitenant environments (for hostability).

However, there are common characteristics to all of them. These characteristics most often include:

- A discrete network architecture that separates production, test, and development environments.
- The isolation of Internet-facing systems from internal systems.
- The configuration of a new or an existing authentication and identity management solution for internal and remote proof-of-identity and access control.
- An effective and efficient redundancy plan — except in the case of single node installations.

Environment Options

Different Unwired Platform environments are designed to support different life cycle stages and whether you are hosting that environment specifically for your enterprise or to host multiple tenants.

This table summarizes the options you can design:

Environment	Purpose	Supports multi-node clusters?	Supports multiple tenants?
Single-node	To install a trial version of Unwired Platform To install all components on a single machine as part of a personal development license.	No, only a single-node installation.	Yes

Environment	Purpose	Supports multi-node clusters?	Supports multiple tenants?
Shared development	To support multiple developers with multiple developing tooling installations on individual computers but share runtime component on the network (that is, servers and runtime databases).	Yes, though typically not needed	Yes, though typically not needed
Fault-tolerant production or pre-production (test) systems	To create a system based on real requirements of an organization and its users.	Yes	Yes

Clustered Environments

Clustering creates redundant Unwired Platform components on your network to provide a highly scalable and available system architecture.

As an organization grows, its need for a scalable IT infrastructure in its environment becomes more critical. Organizations can seamlessly achieve high availability and scalability by adding more or redundant instances of core components. Redundant instances of critical components provide transparent failover. By adding one or more instances of a component, you design a scalable system architecture that simultaneously delivers better performance.

In a production environment, the Unwired Platform deployment typically uses at least one relay server. This is typically not required for shared development environments, or for single-node personal development or trial environments, unless you are testing across a firewall or using a wireless connection from a service provider.

Nodes

A node is a host or server computer upon which one or more components have been installed.

A cluster consists of two or more nodes, that work together as a single, continuously available system to provide application management, device management, and data access to users. Each node on a cluster is a fully functional part of the unwired system. However, in a clustered environment, the nodes work together to provide increased availability and performance.

Unwired Platform network topologies use nodes to show the relationships and connections between hardware and component pairings so that communication channels can more readily be graphed. Those connections show how mobile data moves from one node of a network to the next.

Note: Server nodes in the same cluster should have similar processing, memory, and I/O capability to enable load balancing to occur without significant performance degradation. However, data tier nodes should be more powerful than server nodes.

There are different node strategies you can employ, depending on the environment you are designing:

Strategy	Description	Used for
Single-node	A nonredundant architecture consisting of an Unwired Server and data tier installed on a single host. This strategy is typically used by personal developer licenses.	Personal development or trial installations
2-node	A primitive architecture without load balancing that may optionally use a relay server. The data tier (which includes the CDB, the messaging database, and the monitoring database) is on one node and the application and server tier (which includes Unwired Server, Sybase Control Center, and optionally Afaria) is on another.	Enterprise development or test environments
3-node cluster	A simple redundant architecture with two server tier nodes (which includes Unwired Server, Sybase Control Center, and optionally Afaria) and one data tier node. For an example of this cluster, see *Systems Design > Fault-Tolerant Production Environments > Redundant Architecture Options*.	Entry-level production environments
N+2-node cluster	An optimally redundant architecture with any number of server tier nodes (which includes Unwired Server, Sybase Control Center, and optionally Afaria) supported by a relay server and two data tier nodes. For an example of this cluster, see *Systems Design > Fault-Tolerant Production Environments > Redundant Architecture Options*.	Full-scale or optimized production environments

Cluster Types

Unwired Platform clusters are groups of similar components that work together to service client requests.

A cluster is a parallel or distributed computing system made up of many discrete systems that form a single, unified computing resource. Through clustering, you can partition the system load across redundant Unwired Platform components to design a highly-available mobility system architecture.

There are two key tiers in a cluster: Unwired Servers (the server tier) and runtime databases (the data tier). These tiers may then further be supported by a relay server and perhaps a load balancer depending on the scale of your rollout. However, each tier uses a different type of cluster model:

- Load-balancing server tier clusters – improve the system performance by sharing workloads and requests, and improving the efficiency of Unwired Server services (like synchronization and deployment). Requests initiated from the user are managed by a load balancer, and distributed by relay servers (or some other mechanism) among all the Unwired Server nodes that are part of the cluster. Every node in the Unwired Server cluster scales automatically when another node joins or leaves the cluster.
- Failover data tier clusters – improve the availability of runtime database services to Unwired Servers. Failover clusters have at least one redundant node, which provides data tier services when the primary node hosting the databases fail. The most common size for a failover consolidated database cluster is two nodes, which is the minimum requirement to provide redundancy and thereby eliminate single points of failure.

  Note: The data tier does not need a relay server, since it will not be accessed directly from mobile devices or systems outside the firewall.

Each cluster includes a primary server, and a group of participating servers that are sometimes refered to as secondary servers or slaves:

- The primary server – contains the master copy of the configuration repository for all servers in the cluster. The primary server distributes its configuration to the other servers in the cluster. When a primary server fails, a new primary is elected from the remaining secondary servers.
- The secondary server – gets its configuration from the primary server. Servers must have unique names but are identified as members of the same server farm as the primary server.

Each Unwired Server in a cluster runs on different hosts, so that it can connect to the data tier independently, as well as have its own copy of the system files required for component execution. Sybase recommends that you run each cluster member from its own installation directory.

Load Balancing

Load balancing is a high-availability strategy that prevents any one server from getting overloaded with work, thereby adversely affecting system-wide performance, or even causing the server to fail entirely.

Load balancing primarily supports the server tier. Achieve improved fault-tolerance by balancing loads. Do this by:

- Creating and configuring two or more Unwired Servers in a cluster.
- Using some form of a centralized task distribution mechanism (for example, a relay server).

- Optionally using a third-party load balancer.

Unwired Server Clusters

An Unwired Server cluster provides highly available services to remote-device clients, typically requiring one or more relay servers and an optional load balancer.

When you install Sybase Unwired Platform, a cluster is created automatically, and Unwired Server is added to it. If you are installing the current version on a host that already has a previous version of Unwired Platform installed, the installer allows you to specify a new cluster name to avoid a conflict that might occur when the automatically selected name is used.

By default, a cluster name is the name of the first machine on which you install Unwired Platform. Consequently, each server can be a member of only one cluster. Cluster-configuration properties are saved in the cluster database.

For high availability, at least two servers must be defined for a cluster. However, to create a load-balanced system, many clients then also use a relay server. Without a relay server, clients require custom logic to determine which Unwired Server will directly receive requests. However, a relay-server free environment is not a highly available environment, nor can requests be distributed equally across multiple servers. Consequently, Sybase recommends that you always use a relay server.

See also

- *Relay Server Setup* on page 62
- *Implementing an N+2-node Cluster for Optimal Redundancy* on page 52

Afaria Clusters

You can configure Afaria to use the same relay server as Unwired Server, even if the Afaria Server is installed on a seperate host computer than the Unwired Server tier.

This configuration, which Sybase recommends, lets you seperate the servers, but still share the same relay server installation. Afaria can then also use the same data tier used by Unwired Server.

Each server can be a member of only one cluster. For high availability, there must be at least two servers defined for a cluster.

See also

- *Afaria Setup* on page 82
- *Relay Server Setup* on page 62
- *Afaria Documentation* on page 83

Failover

Failover is another high-availability strategy that allows a secondary server to take over in the event of a fault or failure in the first server, thus allowing normal use to continue. Automatic

failover is a default behaviour of the cluster; it does not require manual intervention or configuration.

Failover primarily supports the Unwired Platform data tier, which includes consolidated database, the cluster database, and the messaging and monitoring database.

Fault-tolerance of the data tier through failover uses a passive/active node configuration where only one node is active at any given moment (that is, if one node fails another standby node then becomes active). Consequently, administrators must ensure that their data storage is protected via a redundant array of independent disks (RAID) cluster.

Database Clusters

Unlike the server tiers for application and device management for Unwired Server and Afaria nodes, the data tier can be clustered to make system data used by the server tier more fail-safe.

When creating a database cluster, administrators must use Microsoft Cluster Server to ensure data fails over and thereby enhances data reliability by ensuring that does not have a single point of failure. Microsoft Cluster Server allows Unwired Platform to fail over the data tier when the current server fails. Both servers (active/failed and passive/takeover) must have a connection to the storage subsystem.

The consolidated database data, along with the Advantage messaging database data, should be stored on a redundant array of independent disks (RAID) cluster. You can use any number of disks in a redundant array. However, the more disks you have in the array, the more fail-safe it is; performance degrades with the more disks you introduce. Sybase recommends that you use a RAID 1 disk array for deployments that require a high degree of fault tolerance and performance from their Unwired Platform data tier.

See also

- *Data Management Overview* on page 158
- *Data Mobility* on page 19
- *Backup and Recovery* on page 234

Shared Development Environments

Mobility projects in medium-to-large organizations usually span multiple development teams, environments, and geographies. In a shared development environment, organizations typically opt for a single-node environment, but may also choose to use domains to partition the environment.

To allow distributed teams to collaborate successfully, you must take an active role in planning the configuration and management of the environment used to develop mobile applications. Your efforts have a direct bearing on the developer's ability to produce quality applications in a timely manner.

To support collaboration, Sybase recommends that developers:

- Share an Unwired Server, so packages can be deployed to a common server. Encourage application developers to develop deployment packages that allow interface designers and business process designers to download the latest versions of the components.
- Place the Unwired Server and data tier on the same node.
- Use a common system data tier for development and testing so that data is centrally managed.
- Optionally divide developers or groups of developers into "tenants", and use domains to partition the server environment. Make each developer (or a representative developer from development groups) a domain administrator so he or she can deploy packages to the shared Unwired Server.
- Ensure that the development environment is similar to, but separate from, the production environment.

  Even if developers create and test their projects locally, they should still test interactions between system components in an environment that mirrors the production environment. Early testing that simulates the production environment, leads to earlier discovery and correction of security and data-tier-related issues.
- Open LDAP, is, by default, installed with Development Edition licenses. The LDAP directory is already populated with required system roles, which allows you to get basic application security running with minimal investment and effort. However, you may want to re-create the application security environment, so development environments and production environments are similar.

See also

- *Chapter 5, Environment Setup* on page 61
- *Multitenant Environments* on page 58

Fault-Tolerant Production Environments

You can design for fault-tolerance by introducing component redundancy in your environment.

For Unwired Platform installations, Sybase recommends a redundant node and cluster strategy for most production environments. A fault-tolerant system supports:

- Load balancing – uses identical Unwired Platform components on a network to balance the number of requests or tasks among these redundant components in parallel.
- Failover – uses identical Unwired Platform components on a network in order to switch to any of the remaining instances in the event of a component failure.

In an Unwired Platform production environment, system design tends to favor load balancing over failover because of the better efficiency it also yields in rather than just ensuring that the system is more reliable.

Redundant Cluster Architectures

Use a redundant architecture to back up Unwired Platform production deployments with multiple secondary resources to create a fault-tolerant environment using clusters on a server or data tier.

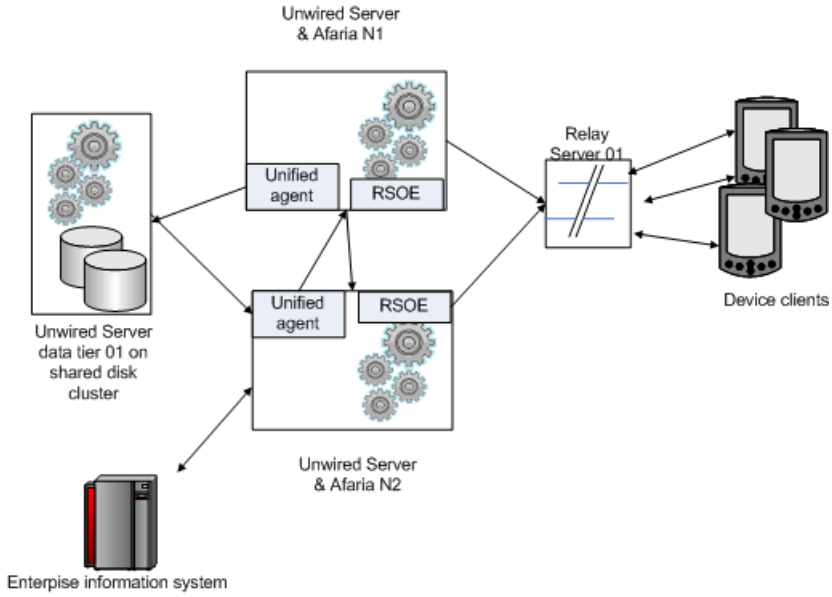
- Hot standby is a method of redundancy in which the primary and secondary server nodes run simultaneously. The data tier is also mirrored to the secondary server in real time, so both systems contain identical information. This is the least expensive redundant architecture; however, Unwired Platform does not support it.
- A simple 3-node cluster is an entry-level redundant architecture used by production deployments.
- An optimal N+2 cluster is the Sybase-recommended production architecture.
- Virtualization allows you to manage system infrastructure on virtual machines. Virtualization allows platform components to run in isolation, but side-by-side on the same physical machine (typically a 64-bit processor with virtual symmetric multiprocessing). Virtualization, while supported in Unwired Platform deployments, is beyond scope of this document.

Simple Redundant Cluster

The simple 3-node cluster is a cluster you can design, and requires minmal hardware investment. Consequently, the simple cluster is an attractive alternative for smaller user environments.

A simple cluster includes:

- A pair of application/device management nodes (Unwired Servers and Afaria) that form the management tier. Each node includes a Unified Agent and a relay server outbound enabler (RSOE). In the illustration below, node 1 (N1) acts as the primary Unwired Server in the cluster and node 2 (N2) acts as the secondary nodes. Both nodes are active.
- One data cluster for Unwired Platform databases (consolidated and messaging databases) to create a single data tier.
- 1 relay server node shared by both server nodes.

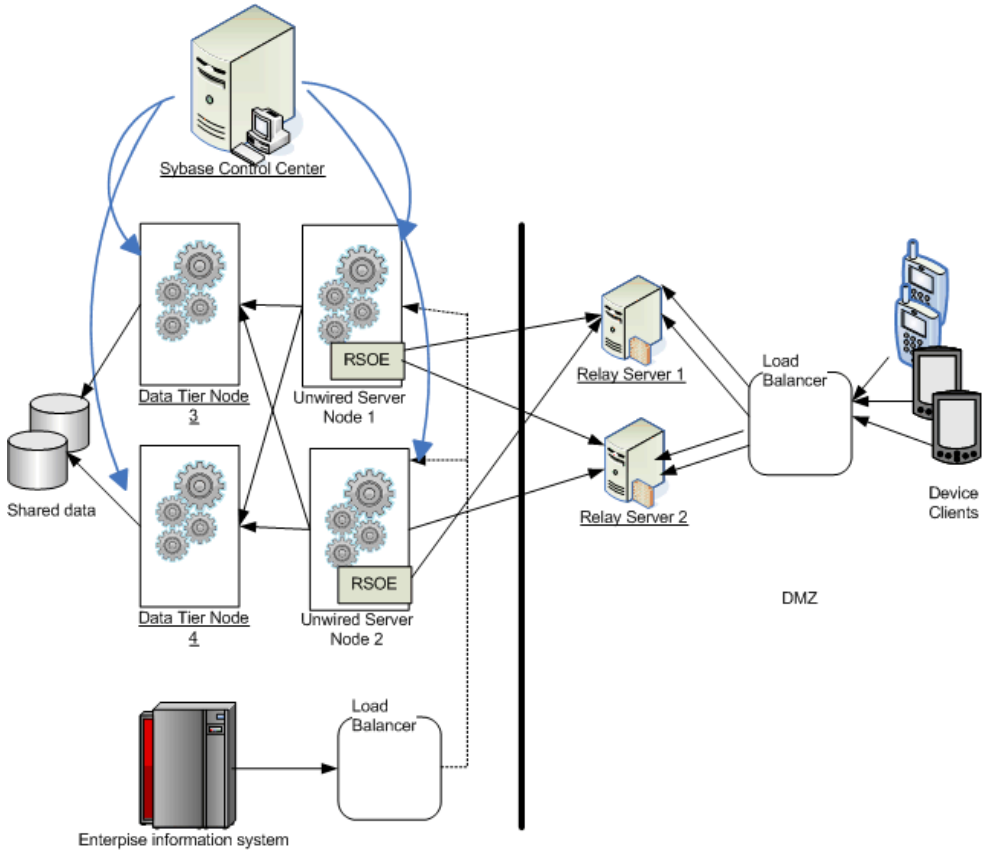


Optimally Redundant Cluster

The optimally redundant cluster is reliable, multi-node cluster and requires a larger hardware investment because of the number of nodes involved. An optimally redundant cluster is the Sybase-recommended solution for larger user environments.

An optimally redundant cluster includes at least four nodes, with seven nodes being optimal:

- Two Unwired Server nodes that form the application management tier. Each node includes a Unified Agent and a relay server outbound enabler (RSOE). In the illustration, node 1 acts as the primary Unwired Server in the cluster and node 2 acts as the secondary node. Both nodes are active.
- (Optional) One Afaria node that forms the device management tier. This node includes an RSOE. In the illustration, node 1 acts as the primary Unwired Server in the cluster and node 2 acts as the secondary node. Both nodes are active.
- (Optional) A load balancer for EIS data change notification requests.
- Two data tiers for Unwired Platform databases (consolidated and messaging databases) to create a redundant data tier over two nodes. Data is shared among them with a disk cluster or RAID array.
- A load balancer with two relay server nodes.



Implementing an N+2-node Cluster for Optimal Redundancy

An N+2 cluster is the recommended cluster for achieving an optimal level of redundancy in your Unwired Platform architecture.

Prerequisites

Sybase recommends that you have Windows 2008 installed on all participating nodes in the cluster, irrespective of whether the node is a runtime server tier node or a data tier node. Windows 2003 is supported on non-cluster installations only.

Once you have set up the environment and deployed components accordingly, perform end-to-end tests that validate the cluster before you make it publically available.

1. *Setting Up the Microsoft Cluster*

   Set up the Microsoft cluster before you install the Unwired Platform data tier.
2. *Setting Up Data Tier Nodes*

Set up a data tier node using the Unwired Platform installer. The data tier node includes the consolidated database (CDB), the cluster database, the monitioring database, and the messaging database. An Afaria database is also installed if you install the Afaria server.

3. *Adding a Generic Service to the Data Cluster*

   Before you can install your runtime server nodes, you must add a generic server resource for the data tier.

4. *Setting Up Runtime Server Tier Nodes*

   Once you have confirmed that the generic service available, you can install and configure the runtime services tier.

5. *Installing Third-party Software*

   If you are using third-party software with Sybase Unwired Platform, copy the third-party software components (DLLs, JARs, and libraries) to each server in a cluster.

6. *Validating the Cluster*

   Once the runtime and data tiers are installed, ensure that these components have been set up correctly.

See also

- *Unwired Server Clusters* on page 47
- *Relay Server Setup* on page 62

Setting Up the Microsoft Cluster

Set up the Microsoft cluster before you install the Unwired Platform data tier.

Setting up the Microsoft cluster may require the assistance of your IT department, especially if you lack the required permissions to complete these tasks.

Note: Perform these steps on all nodes participating in the Microsoft cluster unless otherwise indicated.

1. Ensure failover clustering is added to the Windows 2008 Enterprise Server.

   Failover clustering is required on all nodes to which Unwired Platform data tier components will be installed. Failover clustering is installed, by default, with Windows 2008 Enterprise Server.

   a) Open the Server Manager console.
   b) Click **Features** in the navigation pane to display the **Feature Summary**.
   c) Below Remote Server Administration Tools, look for Failover Clustering Tools.
   d) If Failover Clustering Tools is not listed, click **Add Features** in the right corner of the **Feature Summary** pane and enable this failover clustering feature.

2. Create the MS cluster:

   a) Click **Start > Programs > Administrative Tools > Failover Cluster Management**.
   b) Expand **Management** and start the Create Cluster wizard.

c) On the Access Point for Administering the Cluster page, assign the Unwired Platform cluster name, for example, `SUPdataNode1`.

   Remember this name and use it with the Unwired Platform installer.

d) Finish creating the cluster and exit the wizard

Next

Install the Unwired Platform data tier components before returning to the console to create a generic service for each node.

See also

- *Setting Up Data Tier Nodes* on page 54

Setting Up Data Tier Nodes

Set up a data tier node using the Unwired Platform installer. The data tier node includes the consolidated database (CDB), the cluster database, the monitioring database, and the messaging database. An Afaria database is also installed if you install the Afaria server.

Note: Perform these steps on all nodes participating in the MS cluster unless otherwise indicated.

1. Run the Sybase Unwired Platform installer.

   Use the installer to set up your enterprise deployment license, installation folder, and so on. See *Sybase Unwired Platform Installation Guide*.

2. When prompted to choose the installation type, choose **Cluster Setup** and click **Next**.
3. When prompted to choose the platform features to install, select **Install data tier for your Unwired Platform cluster**, and click **Next**.

   Continue running the installer as documented in *Sybase Unwired Platform Install Guide > Installing the Deployment (Production) Edition > Installing the Deployment Edition in a Cluster > Installing the Data Tier on a Separate Host for a Deployment Installation*.

4. Review the summary and click **Install**.

   The installer creates all required data tier services. For a complete list of services including those names of data services, see *Unwired Platform windows Services*.

   Note: Ensure that the consolidated database server name is the same on all the data tier servers.

5. If required, run the installer on another host to install a failover data node.

   See *Sybase Unwired Platform Install Guide > Installing the Deployment (Production) Edition > Installing the Deployment Edition in a Cluster > Setting Up a Failover Data Cluster*.

See also

- *Unwired Platform Windows Services* on page 261

- *Setting Up the Microsoft Cluster* on page 53
- *Adding a Generic Service to the Data Cluster* on page 55

Adding a Generic Service to the Data Cluster

Before you can install your runtime server nodes, you must add a generic server resource for the data tier.

1. Open the Failover Cluster Management console.
2. Expand <ClusterName>.sybase.com in the navigation pane, then right-click **Services and Applications > Configure a Service or Application** to start the High Availability wizard.
3. Click **Next** until you reach the **Select Service or Application** page.
4. Select data tier service you require, and click **Next**.
   - For the consolidated database (and cluster database), choose **SybaseUnwiredPlatformConsolidatedDatabase**.
   - For the messaging database, choose **Advantage Database Server**.
5. Name the resource.

   You must use this name as the CDB host name when you set up the runtime server cluster.
6. Click **Next** on all other screens to accept the defaults used by the wizard.
7. Click **Finish**.
8. Use the MS Cluster Management console to verify that you started the resource.

   Contact your IT department if this event message is recorded:

   ```
   Event ID 1194: Cluster network name resource <Name>GenSvc failed
   to create its associated computer object in domain "sybase.com"
   for the following reason: Unable to create computer account.
   ```

See also

- *Setting Up Data Tier Nodes* on page 54
- *Setting Up Runtime Server Tier Nodes* on page 55

Setting Up Runtime Server Tier Nodes

Once you have confirmed that the generic service available, you can install and configure the runtime services tier.

Sybase recommends that you use a Windows OS /3GB switch for Windows Server machines with 4GB or more of RAM if you are implementing a cluster in a production environment.

Note: Perform these steps on all nodes participating in the MS cluster unless otherwise indicated.

1. Run the Sybase Unwired Platform installer.

Use the installer to set up your enterprise deployment license, installation folder, and so on. See *Sybase Unwired Platform Installation Guide*.

2. Choose what node you are installing.

   The first node of the cluster determines the cluster name by adopting the host name as the cluster name. For subsequent nodes, you need not provide a name, because each Unwired Server that joins the cluster, assumes the cluster name, with a number that is incremented accordingly. For example, the first server is named SybaseUnwiredPlatformMYHOSTServer1, the second SybaseUnwiredPlatformMYHOSTServer2, and so on.

   Note: If the cluster is to be part of a hosted Relay Server, use only alphanumeric names, since Relay Server does not allow asterisks, spaces, hyphens, or any other special character.

3. When prompted to choose the installation type, choose **Cluster Setup** and click **Next**.
4. When prompted to choose setup type, select either **Install the first server node and connect it to the data tier** or **Install an additional server node and connect it to the data tier**. Provide the Microsoft cluster name (that is, the MS Cluster Access point as the host name of the data tier).

   Continue running the installer as documented in *Sybase Unwired Platform Install Guide > Installing the Deployment (Production) Edition > Installing the Deployment Edition in a Cluster > Installing the First Node for a Deployment Installation* or *Installing Additional Nodes in an Existing Cluster for a Deployment Installation*.
5. Repeat these steps on other hosts until all your server nodes are set up.

Next

In a production environment, you may need to register the server with Sybase Control Center and tune performance properties for the server. See *System Administration > Environment Setup > Server Performance Tuning*.

See also

- *Cluster Administration Overview* on page 126
- *Server Administration Overview* on page 127
- *Adding a Generic Service to the Data Cluster* on page 55
- *Installing Third-party Software* on page 56

Installing Third-party Software

If you are using third-party software with Sybase Unwired Platform, copy the third-party software components (DLLs, JARs, and libraries) to each server in a cluster.

Third-party files is kept separated from Unwired Platform files. There are two reasons for this requirement:

- To isolate third-party from core Unwired Platform files. This can make an upgrade process smoother, since third-party software typically remains untouched.
- To more easily identify the files that need to be applied to a new installation on a new component host machine.

1. Stop Unwired Server.
2. Copy third-party JARs files to `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\lib\3rdparty`.
3. Copy third-party DLLs to `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\lib`.
4. Restart Unwired Server.
5. If you are running a cluster, repeat these steps on all server tier nodes.

See also

- *Setting Up Runtime Server Tier Nodes* on page 55
- *Validating the Cluster* on page 57

Validating the Cluster

Once the runtime and data tiers are installed, ensure that these components have been set up correctly.

Prerequisites

Install sampledb on the first node, and ensure the service has started. sampldb is not installed with a deployment edition license. To install sample db, run:

```
<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Bin
\sampledb.bat install auto
```

End-to-end validation requires that you create a simple MBO and deploy it to Unwired Server. You can use an Unwired Platform tutorial to help you create this package and mobile client.

Task

1. In Unwired Workspace, use a Getting Started tutorial to create a sample MBO and device application that uses data in sampledb.
2. Run the sample application from the emulator.
3. If the synchronization appears to be successful, open the Cluster Management console on your first data tier node, and move the generic service resource you created to the second data tier node.
4. Redeploy your application to the emulator.

Synchronization should work no matter which data tier is used to consolidate platform data from sampledb.

Next
Continue your environment setup and system configuration.

See also
- *Installing Third-party Software* on page 56

Single-Node Environment

A single-node installation does not make use of node or cluster redundancy. In trial or personal development scenarios, you may want to operate Unwired Platform on a single host.

Unwired Platform treats a single-node cluster, or standalone installation, as a cluster of one member.

This type of environment allows for some degree of administrative convenience in a production environment. Typically, however, you use this model to provide a platform for personal development or IT evaluation of Unwired Platform functionality. Considerations for each are summarized by environment type:

- For personal or enerprise development, a single-node installation uses OpenDS by default to authenticate user access for Unwired Platform. It takes, little or no configuration to make this system functional.
- For production deployments, a noAuth provider is enabled by default, which passes all authentication requests. Administrators must configure a provider immediately after installation to ensure that Unwired Server and Sybase Control Center are secured.

See also
- *Chapter 5, Environment Setup* on page 61

Multitenant Environments

Multitenancy allows platform administrators to deploy a single production environment to service multiple client organizations (known as tenants). Multitenancy uses domains, which allow a tenant's administrators to manage Unwired Platform entities within the cluster partition, provided that administration users possess the proper domain administration privileges.

The platform administrator has a complete view of the enterprise and its clusters, including all domains that are created to support the tenancy strategy. However, domain administrators see only the domains to which they have been assigned. Security configurations can be designed and assigned for a particular domain according to the tenant's security requirements. Likewise, domain-specific packages created by the tenant can be deployed to the domain created for him or her.

Do not confuse domains in Unwired Platform with the traditional concept of network domains. In Unwired Platform, a domain is only a namespace used in production environments.

See also

- *Domain Administration Overview* on page 138

Domains

Domains provide a logical partitioning of a hosting organization's environment that achieves increased flexibility and granularity of control in multitenant environments. By default, the installer creates a single domain named "default."

Administrators use different domains within the same Unwired Platform installation. Domains enable the management of application metadata within a partition, including server connections, packages, role mappings, domain logs, and security, so that changes are visible only in the specific domain.

Considerations when implementing domains in a multitenant environment include:

- Create and manage domains using Sybase Control Center from the Unwired Platform administration perspective of Sybase Control Center.
- You can support multiple customers inside the same Unwired Platform cluster.
- You can configure security specifically for individual domains by creating one or more security configurations in the cluster, and then assigning those security configurations to a domain. You can then map the security configurations to one or more packages. A user accessing the package from a device application is authenticated and authorized by the security provider associated with the package.
- Customers may require their own administrative view on their portion of the Unwired Platform-enabled mobility system. By granting domain administration access to your customers, you can allow customers to customize their deployed applications packages and perform self-administration tasks as needed.

The "default" domain

The "default" domain is a special domain where critical runtime configuration artifacts exist. These artifacts include:

- An "admin" security configuration – this security configuration is mapped to the "default" domain and is used to authenticate and authorize administrative users. For this reason, administrators are not allowed to unassign the "admin" security configuration from the "default" domain.
- Consolidated database (CDB) data source connections – for the "default" CDB data source, users can configure the Pool Size property in the "default" domain according to their requirements. This setting allows the maximum number of open connections to the SQL Anywhere database server hosting the CDB.

- Monitor database data source connections – the customer can modify the existing monitoring datasource properties according to their configuration requirements, or create a new monitoring datasource in the "default" domain.

Since these critical runtime-related artifacts are located in the "default" domain, administrators are not allowed to delete this domain. Sybase recommends creating new domains to facilitate tenants according to their application requirements.

CHAPTER 5 Environment Setup

How you set up your environment depends on whether it is a single-node or multinode cluster, development, or production.

Task	Required by
Distribute components to different hosts on the network using the installer in iterations.	• Single-node –not applicable. • Shared development – Required. See *Sybase Unwired Platform Install Guide > Installing Developer Editions*. • Multinode production – Required. See *Sybase Unwired Platform Install Guide > Installing the Deployment (Production) Edition*.
Configure LDAP security for Sybase Control Center.	• Single-node – Optional. OpenDS is typically sufficient. • Shared development – Optional. Only if your environment requires something other than OpenDS. • Multinode production – Required. No default LDAP directory is used with deployment licenses. See *Sybase Control Center > Configure > Configuring Unwired Platform > Security Configurations > Creating a Security Configuration > Security Providers*.
Configure Unwired Server.	• Single-node – Optional.Defaults should be acceptable. • Shared development – Optional. Only if your environment requires something other than the defaults. • Required for the primary server. Configuration is then distributed to secondary servers. See *System Administration > Systems Design > Clustered Environments*.
Set up relay servers and install relay server outbound enablers as required.	• Single-node – Not applicable. • Shared development – Optional. • Multinode production – Required. See *System Administration > Environment Setup > Relay Server Setup*.

Task	Required by
Create and assign security configurations for application security (authentication, authorization, and so on).	• Single-node – Optional. • Shared development – Recommended, to test security features. • Multinode production – Required. See *System Administration > Security Administration > Security Layers > User Security Setup > Security Configurations*.
Create data source connections.	• Single-node – Required. • Shared development – Required. • Multinode production – Required. See *System Administration > Environment Setup > EIS Connections*.
Install third-party software required by some MBO packages	• Single-node – Optional. • Shared development – Optional. • Multinode production – Optional. See *System Administration > Systems Design > Clustered Environments > Implementing an N+2-node Architecture > Installing Third-Party Software* .
Configure the Afaria provisioning environment.	• Single-node – Optional. • Shared development – Optional. • Recommended. Afaria automates many provisioning tasks. See *System Administration > Environment Setup > Afaria Setup*.

Relay Server Setup

Setting up the relay server involves installing it on the network, installing the outbound enabler on each Unwired Platform component host, and configuring both the relay server and its enablers. Sybase recommends that you have a good understanding of the production environment you are supporting.

A relay server environment for Unwired Platform clusters consists of:

- Mobile devices running client applications and services that communicate with Unwired Servers or Afaria Servers running on a corporate LAN.

- (Optional) A load balancer to direct requests from the mobile devices to a group of relay servers. A load balancer is not included with Unwired Platform. However, any load balancer can easily be integrated to request loads distribute load.

  If you use a load balancer, you are setting up what is known as a relay server farm. You can set up a farm with a single relay server, in which case a load balancer is not required; mobile devices can connect directly to the relay server.
- One or more relay servers running in the corporate DMZ.
- Back-end Unwired Servers and Afaria Servers running in a corporate LAN.

  Servers that belong to the same cluster are considered to be in the same relay server farm. A client that makes a request through the relay server must specify the server farm it is targeting.
- One relay server outbound enabler (RSOE) per back-end server node. The RSOE manages all communication between a server and the relay server farm. Its primary function is to initiate an outbound connection to all relay servers in the farm on behalf of the server.
- A configuration file for each relay server. This configuration file defines both a relay server farm and the servers (and their respective RSOEs) that are part of the farm.
- Certificates, if you want to encrypt communications for increased security. See *System Administration > Security Administration > Transport Security Setup > Encrypting Relay Server Communications*.

See also

- *Unwired Server Clusters* on page 47
- *Implementing an N+2-node Cluster for Optimal Redundancy* on page 52
- *Afaria Clusters* on page 47
- *Afaria Setup* on page 82
- *Afaria Documentation* on page 83

Relay Server Documentation

Use the Relay Server documentation to further understand how to use Relay Server to support clustering in an Unwired Platform environment.

This *System Administration Guide for Unwired Platform*, gives some overview and context for administrators who want to install and configure Relay Server in a production environment, as opposed to using the hosted service used by development/test environnments. However, for complete details, read the *Relay Server User Guide*.

If you are viewing this guide as a PDF, you can locate the referenced document at *http://infocenter.sybase.com/*. Go to *Sybase Unwired Platform 1.5.2 > System Administration for Unwired Platform > Environment Setup > Relay Server Setup > Relay Server Documentation* to launch this PDF.

Installing the Relay Server Components to Apache on Linux

After installing Sybase Unwired Platform, extract the relay server component files to the Apache `modules` directory, edit the main configuration file for the Apache HTTP server, and set the environment variables.

Prerequisites

Relay server binaries are included only in the deployment edition of Unwired Platform.

Task

1. Copy `relayserver_linux_x86.tar.gz` from `/modules/relayserver` on the installation media to the host machine. This file is only available on the installation media for the deployment edition of Unwired Platform.
2. Extract the contents to the Apache `modules` directory.

   The file contains:
   - mod_rs_ap_client.so
   - mod_rs_ap_server.so
   - rs.config.sample
   - rshost
   - dblgen11.res
   - libdbtasks11_r.so
   - libdbicudt11.so
   - libdbicu11_r.so
   - libdblib11_r.so
   - dbsupport
3. Copy the generated file from `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\config\rs.config` and save it to the Apache `modules` directory.

   The relay server module expects `rs.config` to be in the same directory as `rshost`.
4. Set the PATH and LD_LIBRARY_PATH environment variables to include the Apache **modules** directory.
5. Edit the Apache `conf/httpd.conf` file:
   a) Add these lines to load the Relay Server client and server modules:

   ```
   LoadModule iarelayserver_client_module modules/
   mod_rs_ap_client.so
   LoadModule iarelayserver_server_module modules/
   mod_rs_ap_server.so
   ```

The client and server modules are invoked using different URLs. The client module explicitly looks for the string **iarelayserver** in the URL path.

b) Add these lines to create a <location> section for the client module:

```
<LocationMatch /cli/iarelayserver/* >
     SetHandler iarelayserver-client-handler
</LocationMatch>
```

c) Add these lines to create a <location> section for the server module:

```
<Location /srv/iarelayserver/* >
     SetHandler iarelayserver-server-handler
     RSConfigFile "/<apache-install>/modules/rs.config"
</Location>
```

The RSConfigFile directive specifies the location of the relay server configuration file, `rs.config`. The `rs.config` file must reside in the same directory where the rshost executable is deployed. For details about configuring this file, see *System Administration > Environment Setup > Relay Server> Configuring a Relay Server in a Multinode Cluster*.

6. Determine if the following environment variables are set globally when Apache spawns a process: $TMP, $TMPDIR or $TEMP.

Make sure the Apache user process has write permissions to the `/tmp` directory location.

- If the environmental variables are set globally, you are finished.
- If these variables are not set globally, or you want the default relay server log file to go in a specific temporary directory, edit the `/<apache-dir>/bin/envvars` file to set and then export TMP.

```
set TMP="/tmp"
export TMP
```

This command sets the environment variable in the shell that Apache creates before it spawns its processes.

Installing the Relay Server Components to IIS on Windows

After installing Sybase Unwired Platform, unzip the relay server component files to the Web site home directory, create an application pool, and enable the relay server Web extensions.

Prerequisites

Relay server binaries are included only in the deployment edition of Unwired Platform. Relay server requires Microsoft Internet Information Service 6.0.

Task

1. Find the appropriate relay server ZIP file located on deployment edition of the installation media in `/modules/relayserver`, and copy it to your relay server system.

There are two versions:

- for 32-bit operating systems use `relayserver.zip`
- for 64-bit operating systems, use `relayserver_x64.zip`

The ZIP file contains the directory `\ias_relay_server` with two subdirectories, `\Client` and `\Server`, which contain the required executables and DLLs for the relay server.

2. Unzip `relayserver.zip` under the Web site home directory that you use for the relay server, typically, `C:\Inetpub\wwwroot`.
3. Create an application pool:
   a) Start IIS Manager Console by opening a command prompt and running:
      ```
      mmc %systemroot%\system32\inetsrv\iis.msc
      ```
   b) Right-click **Application Pools** and create a new application pool, for example, RSOE_AP.
   c) Right-click the application pool and select **Properties** to edit the properties for the outbound enabler application pool. Select the **Performance** tab, and unselect **Shutdown Worker Processes After Being Idle**.
   d) In the **Recycling** tab, unselect **Recycle Worker Processes (In Minutes)**.
4. Enable relay server Web extensions in the IIS Manager Console:
   a) Select the **Directory** tab to edit the properties of ias_relay_server. Set execute permissions to **Scripts And Executables**.
   b) Click **Create under Application Settings**.
   c) Select the application pool you created in step 3.
   d) Under **Web Server Extensions**, allow both `rs_server.dll` and `rs_client.dll` to be run as **ISAPI**.
5. Configure IIS for Unwired Platform device clients to communicate with relay server:
   a) Run IIS admin scripts, which are located in `\Inetpub\AdminScripts`.
   b) Run the following console commands:
      ```
      cscript adsutil.vbs set w3svc/1/uploadreadaheadsize 0
      iisreset
      ```

   If you do not perform this configuration step, you see this error message:

   ```
   Could not connect to the Server. Session did not complete.
   ```

Configuring Relay Server in a Multinode Cluster

Setting up relay server in a multinode environment requires a unique set of steps to ensure configuration details are propagated correctly.

Prerequisites

Ensure you have installed relay server and configured it for your Web server type.

Task

1. On the first node:
   a) Open `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\config\relayserver.properties` and edit property values as explained in the file comments.

      If you are configuring a farm in this file with the `relayserver.webserver.farm_name` and the `relayserver.webserver.token` properties, this farm is used for data orchestration engine (DOE) connections and data change notification (DCN) requests in a cluster.
   b) With Unwired Server running, register the relay server by running `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin\regRelayServer.bat`.

      Registering the relay server also generates `rs.config`. This file is used to configure the relay server.
   c) Restart Unwired Server. The RSOE for this server also starts.
   d) To configure the relay server:

      1. Open `rs.config`, and ensure all servers in the cluster are added to the file. **regRelayServer.bat** generates a configuration file that contains only the details of the server on which it is run — even if there is more than one server registered in cluster.

         For example, when finished, you should have:
         - One `[options]` section
         - One `[relay_server]` section
         - Multiple `[backend_farm]` sections, including one for Unwired Server, one for the messaging synchronisation server, and one for Web server farms). To calculate the total sections, use:

           ```
           (3 * no. of server runtime nodes) = the total number of
           [backend_server] entries
           ```

           For example, if you have two nodes, you would require six `[backend_farm]` sections.

           Define the `[backend_farm]` sections as follows, where the lines prefixed with "#" are additional configuration comments:

           ```
           [backend_server]
               enable   = yes
               farm     = MyServerName.SUPFarm
               id       = MyServerNameSUPServer1
               # uncomment mac if you want relay server to do MAC
               # verification.
               # the value should have only the 1 MAC address RSOE
               # sends to relay server.
              # Look in the RSOE.log file to see what that address is
           ```

```
    # mac      = 00-50-56-c0-00-08!!00-50-56-c0-00-01!!
    # 00-15-c5-3d-e2-01!!00-13-02-c5-8e-f1!!00-ff-b0-b5-
e0-8a
    token    = asd13123bmnsadfas
```

2. Copy `rs.config` to either the `\Servers` directory you created for IIS Web server or the `/Modules` directory you created for Apache Web server.
3. To apply changes, run this command from the directory you copied the `rs.config` file to:

   rshost -f rs.config -q -qc -u

   By default, rshost is installed to *UnwiredPlatform_InstallDir* `\Servers\SQLAnywhere11\Mobilink\relayserver\IIS\Bin32\IAS_relay_server\Server`. For reference details on the rshost utility, see *System Administration > System Reference > Command Line Utilities > Relay Server Utilities, Relay Server (rshost) Utility*.

2. On remaining nodes:
   a) Open `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\config\relayserver.properties` and edit property values as explained in the file comments.
   b) With Unwired Server running, register the RSOE and configuration changes by running `<`*UnwiredPlatform_InstallDir*`>\Servers\UnwiredServer\bin\regRelayServer.bat` with Unwired Server running.
      Use this syntax:
      ```
      regRelayServer.bat install auto
      ```
      This file generates:
      - `<`*UnwiredPlatform_InstallDir*`>\Servers\UnwiredServer\config\rs.config`
      - `startrsoe.bat` `stoprsoe.bat` in `<`*UnwiredPlatform_InstallDir*`>\Servers\UnwiredServer\bin\`.

      After `startrsoe.bat` and `stoprsoe.bat` are generated, the Start Unwired Server desktop shortcuts and menu commands also start and stop the RSOE.

      Note: These various `.BAT` files are never modified while registering relay server. They have a condition to check if relay server is configured.
   c) If relay server is configured with the generated `rs.config` file, restart Unwired Server. The RSOE for this server also starts. If a file other than the generated `rs.config` is used, RSOEs will have errors.
3. For messaging-based application devices, register the device using relay server connection parameters:
   a) In Sybase Control Center, create a registration template using exported values.

For details, see *Sybase Control Center Online Help> Configure > Configuring Unwired Platform > Device Users > Devices*.

b) Set up the device messaging application settings to connect the relay server.

Configure properties of the same name as follows:

- User Name – any valid user name that has been registered with the messaging server.
- Server Name – `MyRelayserver.MyCompany.com`
- Server Port – `80`
- Company ID – the messaging server farm ID. For example, **SUPServer.SUPMessagingFarm**.
- Activation Code – a matching relay server code.

4. For replication-based synchronization application devices, the application developer can embed the relay server connection info in the application, or the administrator can provide the setup values to the application user, thereby allowing him or her to configure the device application independently.

See also

- *Relay Server (rs.config) Configuration File* on page 336

Copying a Required Relay Server Files

After you have installed relay server, you must ensure its files get copied to a specific location. If the files are copied incorrectly, the device cannot connect to the relay server.

Sybase recommends that you set a system PATH variable for the Web server that hosts the relay server. For example, on IIS this might be `C:\Inetpub\wwwroot\IAS_relay_server\Server`.

Configuring Relay Server to Run as a Windows Service

Sybase recommends that you set up the relay server as a Windows service to ensure that it starts automatically when the host is started.

1. Enter the following at the command line, substituting all parameter values to match your configuration:

```
dbsvc -as -s auto -w SUPRelayServer "C:\inetpub\wwwroot
\ias_relay_server\server\rshost.exe" -q -qc -f c:\inetpub
\wwwroot\ias_relay_server\server\rs.config -o c:\Sybase
\logs\rs.log
```

This command configures the relay server host process (rshost.exe) as a Windows service. By default, the dbsvc utility is installed to `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32\`.

Note: Ensure that the path specified with the -o option exists; otherwise the command fails.

2. Start or stop the rshost service.

   From the Windows Services control panel:

   - Locate the **SQL Anywhere - SUPRelayServer** service.
   - To start or stop the service, right-click the service and choose the corresponding command.

   From the command prompt:

   - Change to `C:\inetpubs\wwwroot\ias_relay_server\Server`.
   - To start the service, enter `dbsvc.exe -u SUPRelayServer`.
   - To stop the service, enter `dbsvc.exe -x SUPRelayServer`.
   - To uninstall the service, enter `dbsvc.exe -d SUPRelayServer`.
   - To update the rshost with the latest relay server configuration, enter `rshost.exe –f rs.config –u`.

See also

- *RSOE Service (rsoeservice) Utility* on page 299

Using the Sybase Relay Server Hosted Service

Subscribe online to the Sybase Relay Server Hosted Service (SRSHS), and configure your environment with information provided during subscription, rather than installing a relay server on your network.

1. Register Unwired Server with the relay service.
   a) Log in to *https://relayserver03.sybase.com/ias_relay_server/account/login1.php*.
   b) From the navigation bar on the left, click **Manage Account** .
   c) Click **Add New MobiLink Farm** enter the following property values, then click **Create Farm**:
      - **Farm Name** – a farm name. Sybase recommends SUPFarm.
      - **Server Name** – the configured Unwired Server name. For example, host1SUPServer. You can find the name checking the server properties in Sybase Control Center. Alternatively you can check the sup.host value in the `sup.properties` file. This file is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer`.
   d) Click **Add New iAnywhere Mobile Office Farm**, enter the following property values, then click **Create Farm**:
      - **Company Name** – a company ID used to create the server farm. Sybase recommends SUPMessagingFarm.

- **Server Name** – the configured messaging synchronization service name. For example, host1.

e) Click **Configuration Instructions**. You need these values to complete configuration values in subsequent steps.
In a replication-based synchronization environment, the exported values should look similar to:

```
-f SUPServer.SUPFarm
-id host1SUPServer
-t 91ae78ca50b90b02cef494c869c6
-cr "host=relayserver.sybase.com;https=1;port=443"
-cs "host=localhost;port=80"
```

Note: The farm ID (-f) and Back-end server ID (-id) are a case-sensitive property values. Therefore, they must exactly match the value defined for the relay server outbound enabler (RSOE) with relayserver.farm_name property (described in later steps). Otherwise, the connection fails.

The messaging-based synchronization environment values are:

```
-f SUPServer.SUPMessagingFarm
-id host1
-t 8361c7e7783527e932617c49467f
-cr "host=relayserver.sybase.com;https=1;port=443"
-cs "host=localhost;port=80"
```

2. Configure the Unwired Server and messaging synchronization service to use an RSOE. The RSOE allows these servers to connect to the hosted relay server.

a) Use a text editor to open *<UnwiredPlatform_InstallDir>*`\Servers\UnwiredServer\config\relayserver.properties`, and modify the values in this file to match the values exported in the previous step.
For example, in a replication-based synchronization environment:

```
cluster.name = host1SUPServer
relayserver.host = relayserver.sybase.com
relayserver.http_port = 80
relayserver.https_port = 443
relayserver.protocol    = http
relayserver.farm_name    = SUPServer.SUPFarm
relayserver.token    =
91ae78ca50b90b02cef494c869c6
```

Note: You can secure the connection by setting the protocol to HTTPS.

For the messaging service, you must also modify:

```
#-------------------
# Message Server details
#-------------------
relayserver.msg.farm_name = SUPServer.SUPMessagingFarm
relayserver.msg.token    = 8361c7e7783527e932617c49467f
```

b) With Unwired Server running, register the RSOE and configuration changes by running *<UnwiredPlatform_InstallDir>*`\Servers\UnwiredServer\bin\regRelayServer.bat` with Unwired Server running.

Use this syntax:

```
regRelayServer.bat install auto
```

This file generates:

- *<UnwiredPlatform_InstallDir>*\Servers\UnwiredServer\config\rs.config
- `startrsoe.bat` `stoprsoe.bat` in *<UnwiredPlatform_InstallDir>*\Servers\UnwiredServer\bin\.

After `startrsoe.bat` and `stoprsoe.bat` are generated, the Start Unwired Server desktop shortcuts and menu commands also start and stop the RSOE.

Note: These various `.BAT` files are never modified while registering relay server. They have a condition to check if relay server is configured.

c) Restart Unwired Server so that the RSOE starts. If both the Unwired Server and messaging server successfully connects to Web-hosted relay server, you should see entries in the console window similar to this:

```
I. 2009-10-28 13:49:26. <OEMaster> Successfully connected to
Back-end server: easd:5001
I. 2009-10-28 13:49:26. <OEHost> Mac address:
00-13-72-19-6d-97!!00-13-72-19-6d-97!!
I. 2009-10-28 13:49:27. <UpChannel-0000> Successfully connected
to relay server: RELAYSERVER03.sybase.com:80
I. 2009-10-28 13:49:27. <UpChannel-0001> Successfully connected
to relay server: RELAYSERVER04.sybase.com:80
I. 2009-10-28 13:49:27. <DnChannel-0000> Successfully connected
to relay server: RELAYSERVER03.sybase.com:80
I. 2009-10-28 13:49:27. <DnChannel-0001> Successfully connected
to relay server: RELAYSERVER04.sybase.com:80
```

3. For messaging-based application devices, register the device using relay server connection parameters:
   a) In Sybase Control Center, create a registration template using exported values.
      For details, see *Sybase Control Center Online Help> Configure > Configuring Unwired Platform > Device Users > Devices*.
   b) Set up the device messaging application settings to connect the relay server.

   Configure properties of the same name as follows:
   - User Name – any valid user name that has been registered with the messaging server.
   - Server Name – `MyRelayserver.MyCompany.com`
   - Server Port – `80`
   - Company ID – the messaging server farm ID. For example, **SUPServer.SUPMessagingFarm**.
   - Activation Code – a matching relay server code.
4. For replication-based synchronization application devices, the application developer can embed the relay server connection info in the application, or the administrator can provide

the setup values to the application user, thereby allowing him or her to configure the device application independently.

Next

Configure the RSOE to run as a Windows service.

Configuring RSOE to Run as a Service

Configure the relay server outbound enabler (RSOE) to run as a service and start automatically; otherwise you must start and stop it manually, with the Windows Services control panel.

Prerequisites

You must preregister all Unwired Platform servers and synchronization services with the Sybase-hosted relay service to get the required `startrsoe.bat` and `stoprsoe.bat` files. See either *Configuring Relay Server in a Multinode Cluster* or *Using the Sybase Relay Server Hosted Service* tasks for details.

Task

1. Start the relay server outbound enabler as a Windows service.

   ```
   cd <UnwiredPlatform_InstallDir>\Servers
   \SQLAnywhere11\Bin32
   dbsvc -as -s auto -w rsoe "<UnwiredPlatform_InstallDir>
   \Servers\SQLAnywhere11\BIN32\rsoe.exe" -cr
   "host=RELAYSERVER.sybase.com;https=1;port=443" -cs
   "host=myUnwiredServer;port=5001" -f "abc.supqadmz2msg" -id
   "SUPQADMZ2" -t "Token_0" -v "3" -o "D:\temp\rsoe.log"
   ```

   Use the same information used to configure the Sybase-hosted relay server. Substitute all parameter values for those shown here to match your configuration, where
   - -cr is the relay server connection information
   - -cs is the Unwired Server connection information
   - -f is the farm ID used
   - -t is the authorization token
   - -o is the location of the log output
2. To start or stop the rsoe service independently from other platform services:

   From the Windows Services control panel:
   - Locate the **SQL Anywhere - rsoe** service.
   - Ensure that the Startup Type is set to Automatic.
   - Right-click the service name and select the corresponding menu option to start or stop the service.

   From the command prompt:

- Change to `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\Bin32`.
- To start the service, enter `dbsvc.exe -u rsoe`.
- To stop the service, enter `dbsvc.exe -x rsoe`.
- To uninstall the service, enter `dbsvc.exe -d rsoe`.

Updating the Relay Server Configuration File

Use the rshost utility to propagate changes in the relay server configuration file to a running instance of the relay server running on the same machine. Repeat this process on each node on which a relay server is installed.

1. Stop the relay server.
2. Change the configuration file as required.
3. From the command line, run the utility with this exact syntax:

   rshost -u -f <filename>

   The -u option instructs rshost to detect configuration updates and apply them to the currently running relay server. The -f option is used to specify the new configuration file.
4. Restart the relay server.
5. Repeat these steps for all relay servers.

Configuring Relay Server Timeouts for DCNs

Prevent data change notifications (DCNs) from timing out in a cluster, by configuring relay server outbound enabler (RSOE) properties.

Increase timeouts for the RSOE, application pooling, and Web server connections.

- Increase the RSOE timeouts in the RSOE configuration file. The default value of the `timeout configuration` property, 100, is insufficient for DCN use in a cluster, and may trigger errors similar to this on a client:

  ```
  ERROR|11/12 07:37:59 A|user62|Error in Sync :
  System.Net.WebException: The operation has timed out
    at System.Net.HttpWebRequest.GetResponse()
    at PerfTest15.Utilities.SendDCN(Int32 startOrderId, Int32
  numUpdates)
    at PerfTest15.Utilities.ExecuteTest()
  ```
- Increase the application pooling timeouts and Web server connection timeouts in your Web server configuration manager.

1. Open the RSOE configuration file and increase `timeout configuration` from 100 to infinite.

   See the *Relay Server Users Guide* PDF that is accessible from *System Administration Guide > Environment Setup > Relay Server Setup > Relay Server Documentation*.

2. Open the configuration manager for your Web server, and modify the application pooling property for the RSOE.

   For example, for IIS, the manager is called Inetmgr.
   - Configure application pooling for `RSOE_AP` by enabling the **Enable Rapid-fail Protection** property, and then setting the **Time-Period** property to a value higher than 5 minutes.
   - Configure connection timeouts for the **Web Site** by setting a **Connection Timeout** value higher than 120 seconds. Ensure that **Keep Alives** is selected.

   See the Web server documentation for details.

See also
- *Encrypting DCN Connections* on page 101
- *Data Change Notifications* on page 163

Server Performance Tuning

Consider server configuration properties when tuning performance. You can tune performance immediately after installing Unwired Server. Or you can tune it as part of regular system administration duties in an ongoing capacity.

Ensure that all applicable performance-related settings are properly configured:

- Java heap size – when the garbage collection log indicates high activity and the JVM memory allocation pool heap size remains close to the maximum, you may need to reconfigure the maximum and minimum heap sizes. You can enable garbage collection tracing by setting the "JVM options" property. Set appropriate maximum and minimum heap sizes in the **Server Configuration** node of Sybase Control Center. The default minimum heap size is 64MB and the default maximum heap size is 256MB. See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Properties > General Server Ports > Configuring System Performance Properties*.
- Java thread stack size – if you see a StackOverflowError, you may need to increase the default stack size from 400KB. Set an appropriate Java thread stack size in the **Server Configuration** node of Sybase Control Center. Alternatively, you can also decrease the stack size to save more memory. See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Properties > General Server Ports > Configuring System Performance Properties*.
- Messaging-based synchronization (MBS) – if there are large numbers of MBS clients and response times or queue counts are high, consider the inbound and outbound messaging queue counts. These numbers control the number of message queues used for incoming and outgoing messages between the messaging-based synchronization application and the server. View messaging queue statistics in the **Monitoring** node of Sybase Control Center to calculate the appropriate values and enter these in the **Server Configuration** node of

Sybase Control Center. See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Properties > General Server Ports > Configuring System Performance Properties*.

- Replication-based synchronization (RBS) – if RBS client application performance does not meet the desired level and the CPU or memory settings are not causing this issue, consider increasing the synchronization thread count and cache size for the RBS server. Choose an appropriate synchronization cache size and thread count in the **Server Configuration** node of Sybase Control Center. See *Sybase Control Center online help> Configure > Configuring Unwired Platform > Unwired Server > Server Properties > Replication > Configuring Replication-Based Synchronization Properties*.
- Consolidated database (CDB) – if the CDB server is slow, consider increasing the CDB thread count to allow for more parallel threads to serve incoming requests from the Unwired Server nodes. See *Changing the Consolidated Database Server Thread Count*. Also consider tuning the number of client connections that are saved in the CDB connection pool. Configure this in the 'default' domain by modifying the 'default' data source pool size property in Sybase Control Center. Repeat this change on all nodes in the cluster.
- Data change notification (DCN) listener threads – if there are a large number of concurrent requests and processing times for those requests do not meet requirements, consider increasing the number of DCN listener threads. Set an appropriate number of listener threads in the **Server Configuration** node of Sybase Control Center. See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Properties > General Server Ports > Configuring Communication Port Properties*.
- Server host memory – if you are running Unwired Server on a host with 4GB of memory, by default, 2GB of memory is allocated to applications, such as Unwired Server. If performance issues persist after you have adjusted the aforementioned settings, consider starting Windows with a "/3GB" switch. See *http://technet.microsoft.com/en-us/library/bb124810(EXCHG.65).aspx*.

Note: For Java min/max heap size, and Java thread stack size properties, once you make the change you must restart Unwired Server. For configuration details, see topics in *System Administration > Systems Administration > Server Environment Administration* . For recommendations, see *Configuring System Performance Properties*.

See also

- *Configuring System Performance Properties* on page 133
- *Applying Performance Tuning Changes if Unwired Server is a Service* on page 134
- *Changing the Consolidated Database Server Thread Count and Pool Size* on page 79

Database Setup

Setting up a database is only required when you are not installing a new consolidated database (CDB) or monitoring database in your production environment.

There are three scenarios that Unwired Platform supports:

- Use only what is installed with the Unwired Platform installer – This is the simplest and recommended scenario; no additional setup is required. You can begin using the databases immediately.
- Use an existing SQL Anywhere installation, but create new database files based on Sybase specifications – This requires that you use a combination of dbinit to initialize the new CDB correctly, and batch scripts that setup the schema for the monitoring database.
- Use an existing SQL Anywhere installation, but use existing database instance – Can be the most risky implementation and of the three is the least recommended option. You must still use the batch scripts to setup the monitoring database schemas; for the CDB, you must use RTRIM to avoid getting incorrectly formatted data on the device. See the SQL Anywhere documentation about using the RTRIM function.

1. *Initializing a New Consolidated Database*

   If you are experiencing incorrect string comparisons, you may need to initialize a new database to ensure that comparisons occur correctly. Incorrect string comparisons only occur when you either use a SQL Anywhere database that has not been installed and set up with the Unwired Platform installer.
2. *Setting Up an Existing Database for Monitoring*

   You can use any SQL database provided that the existing version number of that existing database matches the version required by Unwired Platform.
3. *Changing the Consolidated Database Server Thread Count and Pool Size*

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

Initializing a New Consolidated Database

If you are experiencing incorrect string comparisons, you may need to initialize a new database to ensure that comparisons occur correctly. Incorrect string comparisons only occur when you either use a SQL Anywhere database that has not been installed and set up with the Unwired Platform installer.

Initialize a database using the dbinit utility. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32`. For details, see the SQL Anywhere dbinit documentation at *http://dcx.sybase.com/index.html#1100en/dbadmin_en11/dbinit.html*.

By default, SQL Anywhere creates a database treating a trailing blank as any other character in comparisons during collation. To avoid treating trailing blanks as significant characters, a CDB should be created with the "-b" option that directs SQL Anywhere to ignore trailing blanks in any comparison.

Run dbinit using this syntax: `dbinit -z UTF8 -zn UTF8 -b`

Where:

- -z UTF8 sets the collation sequence data as UTF8.
- -zn UTF8 sets the collation sequence for sorting and comparing to UTF8.
- - b sets up blank padding.

See also

- *Setting Up an Existing Database for Monitoring* on page 78

Setting Up an Existing Database for Monitoring

You can use any SQL database provided that the existing version number of that existing 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\SQLAnywhere11\BIN32`.
   - For SQL Anywhere databases:
     - `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.
   - For ASE databases:
     - `init-monitoring-tables-ase.sql` – sets up the Adaptive Server database with a general monitoring schema.
     - `ASE_MONITORING_DDL.sql` – sets up the Adaptive Server database with a cache monitoring schema.

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

See also

- *Monitoring Database* on page 14
- *Status and Performance Monitoring* on page 186
- *Monitoring Database Schema* on page 339
- *Initializing a New Consolidated Database* on page 77
- *Changing the Consolidated Database Server Thread Count and Pool Size* on page 79

Changing the Consolidated Database Server Thread Count and Pool Size

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

Sybase has identified two scenarios.

See also

- *Update Properties (updateprops.bat) Utility* on page 318
- *Server Performance Tuning* on page 75
- *Setting Up an Existing Database for Monitoring* on page 78

When data-tier is on remote host in a cluster environment

1. On the data-tier node, remove the service by running:

   ```
   <UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin
   \install-sup-sqlany11.bat remove
   ```

2. To change the thread count, edit the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin\install-sup-sqlany11.bat` to modify the CDB_THREADCOUNT property.

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

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

   ```
   set CDB_THREADCOUNT=200
   ```

3. To change the maximum pool size, edit the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\sybase\djc\sql\DataSource\default.properties` to modify the maxPoolSize property.

4. Recreate the service again with the correct thread count by running:

   ```
   <UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin
   \install-sup-sqlany11.bat install [auto|manual]
   cdb_servername cdb_serverhost cdb_serverport
   [shared_data_path service_username service_password]
   ```

When data-tier is on the same host as Unwired Server in a personal or evaluation install

1. On the Unwired server node with data-tier installed, update properties by running:
   ```
   <UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin
   \updateProps.bat -nv "cdb.threadcount=200"
   ```
   Choose an appropriate server threadcount according to your cluster performance requirements.
2. To change the maximum pool size, edit the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\ybase\djc\sql\DataSource\default.properties` to modify the maxPoolSize property.
3. Stop Unwired Server services.
4. Reconfigure Unwired Platform services by running:
   ```
   <UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin
   \configure-mms.bat <clusterName>
   ```

EIS Connections

Prepare Unwired Server to establish connections with different drivers and connectors, especially to non-Sybase data sources like SAP.

Unwired Server includes and uses the appropriate drivers for Sybase databases like Adaptive Server and SQL Anywhere databases. It also has the necessary drivers to connect to Web services.

However, for non-Sybase data sources such as SAP, you must set up Unwired Server to establish connections to them.

Depending on the type of connection required, certain files need to be copied to Unwired Server directories. Repeat this process on each server in the cluster.

See also

- *EIS Connection Management Overview* on page 146
- *EIS Data Source Connection Properties Reference* on page 277

Preparing Unwired Server to Connect to JDBC Databases

Connections to Oracle, DB2, and SQL Server databases must be prepared for by downloading the appropriate JDBC driver and installing them to the correct location.

Once you have placed drivers in the correct location for Unwired Server, you can connect to JDBC data sources correctly.

1. Shut down Unwired Server.
2. Download the driver.

JDBC driver	URL
Oracle	*http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html*
DB2	*http://www-306.ibm.com/software/data/db2/express/download.html*
SQL Server	*http://msdn.microsoft.com/en-us/data/aa937724.aspx*

3. Place the JDBC driver in the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\lib\3rdparty` directory.
4. After copying the driver file, restart Unwired Server.
5. Repeat these steps on each server in the cluster.

Preparing Unwired Server to Connect to SAP using Java Connectors

Sybase recommends that you make all changes concurrently in a distributed environment for development and production installations of Unwired Platform.

Prerequisites

Before you can access the SAP Web site, you must have an SAP account.

These steps describe the common tasks for setting up an SAP in the Unwired Server runtime.

Task

1. Go to the SAP Web site at *http://service.sap.com/connectors* and download the latest SAP Java Connector, for example, sapjco-ntintel-2.1.8.
2. Unzip the file, which contains:
   - `sapjco.jar`
   - `librfc32.dll`
   - `sapjcorfc.dll`
3. Shut down Unwired Server.
4. Copy (or replace, if it already exists) `librfc32.dll` into `<WINDOWS_HOME>\System32`.
5. Copy (or replace, if it already exists) `sapjcorfc.dll` into `<SUP_HOME>\Servers\UnwiredServer\lib`
6. Copy (or replace, if it already exists) `sapjco.jar` into `<SUP_HOME>\Servers\UnwiredServer\lib\3rdparty`.
7. After copying the driver file, restart Unwired Server.
8. Repeat these steps on each server in the cluster.

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.

Device User Credentials and EIS Connections

Administrators should be aware that developers may map a connection's user name and password properties to system-defined "username" and "password" personalization keys, respectively. This creates a new connection for each client and the connection is established for each request (no connection pooling).

During deployment of this kind of MBO package, administrators must map the design-time connections to the production EIS server connections. However, the username and password for the selected server connection are replaced by the user name and password propagated from the device application via the personalization key.

Afaria Setup

The type of setup required in large part depends on whether you are doing a new install with the Unwired Platform installer or using an Afaria installation that pre-exists.

Initial setup of a Afaria requires that you use:

Task	New Install	Existing Install
The Unwired Platform installer – to install Afaria components on the same or different server node as Unwired Server. Typically Afaria should be installed to a separate node as scaling requirements differ. See *Sybase Unwired Platform Install Guide*.	Yes	Not applicable
Sybase Control Center – to register the Afaria server and thereby launch the Administrator from this tool. See *Sybase Control Center online help> Get Started> Get Started with Unwired Platform > Opening the Afaria Administration Console*.	Yes	Yes
Afaria Administrator – to configure Afaria correctly to Provision Unwired Platform Devices. See *System Administration for Sybase Unwired Platform > System Administration > Device and User Management > Device Provisioning*.	Yes	Yes
Various Unwired Platform setup scripts – to install and configure the OTA Deployment Center. See the *Afaria Install Guide*.	No	Yes
Use a relay server with Afaria servers See *Afaria Install Guide*.	Required if needed by Unwired Server. Optional, otherwise.	Required if needed by Unwired Server. Optional, otherwise.

See also

- *Afaria Clusters* on page 47
- *Relay Server Setup* on page 62
- *Afaria Documentation* on page 83
- *Afaria Provisioning and Mobile Device Management* on page 177
- *Afaria Licenses* on page 243

Afaria Documentation

Use the Afaria documentation for information about using Afaria for device provisioning in an Unwired Platform environment.

The *System Administration Guide for Unwired Platform* gives some overview and context for administrators who want to use Afaria to provision devices that act as clients to Unwired Server. However, Afaria documentation describes in detail the features set of this product.

See these Afaria PDFs: *Installing Afaria*, *Afaria Reference | Platform*, and *Afaria Reference| Components*. These documents are available at *http://infocenter.sybase.com/*. Go to *Sybase Unwired Platform 1.5.2 > System Administration for Unwired Platform > Systems Design > Afaria Setup > Afaria Documentation*.

You can also review the onlinle help associated with Afaria Administrator. The documentation can be located on the Afaria server host: `<UnwiredPlatform_InstallDir>\Servers\AfariaWebUI\Help`.

See also

- *Afaria Clusters* on page 47
- *Afaria Setup* on page 82
- *Relay Server Setup* on page 62

Installing Additional Afaria Components

Install additional Afaria components from the Sybase Unwired Platform installation media.

1. Insert the Sybase Unwired Platform installation media (Deployment Edition).
2. Open the `\modules\afaria\Extras` directory, which contains executables and support files for Afaria components.

Folder	Component description
DeploymentCenter	Scripts that support OTA (over-the-air) deployment.
Packager	Contains the installation files for Software Packager.
RemoteControl	Contains third-party files for the Afaria Remote Control solution, with subdirectories for supported devices: Pocket PC, Pocket PC Windows Mobile, and Smartphone Windows Mobile. Includes product installation and product documentation. Remote Control allows an administrator to view and manipulate a client's environment, typically for troubleshooting purposes.
SnapIns	Contains supporting components for the Afaria SMS Integration Suite setup program.
SWMTuner	Contains the primary installation files for Afaria Software Manager Tuner, which enables you to customize a Windows installer package.
TuneInst	Contains supporting files used during the Software Manager Tuner installation.

CHAPTER 6 Security Administration

Because Unwired Server coordinates data between enterprise information system data sources and device clients, it is likely that information is proprietary and confidential. Therefore, the data needs protection.

Unwired Platform has several features to assist in building a secure end-to-end environment. Understand what features exist before you begin configuring and implementing various security mechanisms.

Security Layers

The platform administrator is responsible for most aspects of security, however application security can be managed by domain administrator.

Security in Unwired Platform can be separated into distinct areas:

- Transport security – secures client/server communications by using digital certificates and public-key cryptography. Public-key cryptography is a widely used approach that secures communication between source and destination components as data is transferred across public or private networks.
- User security – authenticates and authorizes user or administrator access by using the services of a configured provider listed in a security configuration. Once authenticated, access is given to perform operations with a package. However, application data access then further depends on the logical role assigned to the MBO and operations, and its mapping to a role/group in the provider.

  Unwired Server relies on user entries that are stored in a existing directory like LDAP. When a user requests data in the device application, the application uses Unwired Server to authenticate the user and authorize access to specific resources with configured providers.
- Network security – secures components behind a firewall using a proxy. Unwired Platform components are usually installed in a private network, thereby isolating it from external network by one or more intermediary networks. Typically, this involves separating Unwired Servers from clients, either because of firewall policies or address space concerns. You then use a proxy server outside of the firewall (for example, the relay server) to open ports that allow secure data passage through the firewalls to Unwired Platform components.

  Most Unwired Platform installations will use the Relay Server as the network proxy in production scenarios. The Relay Server proxy tunnels between network zones; you can chain multiple servers together to allow for multiple hops. This network architecture allows the Unwired Server to connect and communicate with remote client devices across

multiple network boundaries (for example, the Internet, wireless network, or even other private networks).

- Data security – ensures that data is kept safe, and that access to it is controlled to protect data and keep it private. Data security has two sides:
  1. Data security – can be secured in platform databases by changing the DBA password and enabling encryption. Client-side databases can be encrypted using the Unwired Platform API.
  2. Device security – the ability to lock or wipe the device of enterprise data if the device is lost or stolen. You can use Afaria for this purpose.See *System Administration > Environment Setup > Afaria Setup*.

Together, these levels provide multitiered and complete protection in an Unwired Platform environment — especially when used with a relay server. Relay server (as opposed to a typical proxy server) allows secure data communication across network zones without opening holes in firewalls. This makes relay server more secure than the average proxy server counterparts. See *System Administration > Environment Setup > Relay Server Setup*.

Transport Security Setup

Device client communications, back-end server communications, and Unwired Platform system server communications can all be encrypted.

To fully secure the network communications of all components in a Unwired Platform enabled production environment, you need to enable encryption in the following areas of an Unwired Platform-enabled production environment:

- Front-line data exchange: Encrypting replication based synchronization data that is transfered between device clients and Unwired Platform components. For RBS device clients, you also have to install a keystore with the HTTPS server's keys on the client as well. For MBS device clients, data is always encrypted and does not require any further configuration.
- Back-end server data exchange: Sending data change notifications by using a secure interface that notifies Unwired Server of changes to EIS data.
- Administrative data: Encrypting data exchanged among Sybase Control Center and Unwired Server.

Protocol and Component Reference

Review this summary of encrypted protocols used in Unwired Platform, and how each component is configured to enable encryption.

Transport stream	Encrypted protocols supported	Configuration
Unwired Server and replication-based client applications. **Note:** Messaging data is exchanged using a proprietary encryption over HTTP and there is no need to use SSL.	HTTPS using SSL	Generate the certificate, copy the file to the Unwired Server directory, then configure HTTPS in Sybase Control Center in the Unwired Server replication synchronization configuration properties.
Data change notifications (DCNs) between an enterprise information server(EIS) and Unwired Server	HTTPS using SSL	(EIS developer) use HTTPS to construct and send DCN requests to the listener. (Administrator) generate certificate, importing certificate into server-side keystore, then using Sybase Control Center to configure an HTTPS listener for DCNs in the Unwired Server configuration properties.
Unwired Server and Sybase Control Center	IIOPS using SSL	Generate the certificate, import the certificate into the Unwired Server key store, import the public certificate into the Sybase Control Center key store, configure IIOPS in Sybase Control Center, in the Unwired Server configuration properties, then update the Sybase Control Center server resource properties to use the IIOPS port and secure mode.
Unwired Server and relay server	HTTPS using SSL	Configuring the Web server used for the relay server (IIS or Apache) to use a secure port, then setting the correct properties (for example, relayserver.protocol, relayserver.trusted_certs) in the `relayserver.properties` file and configuring relay server.

Transport stream	Encrypted protocols supported	Configuration
Device clients and relay server	HTTPS using SSL	The `relayserver.properties` file, but only if mutual authentication is required. In most cases, LDAP authentication, or some other authentication provider is sufficient.
Device clients and Afaria server	HTTPS or XNETS using SSL	Afaria Administrator for client communication properties.

Note: In addition to configuring each component to use the correct protocol and port, you must secure all of your generated server/client certificates.

Security Profiles

In Sybase Control Center, transport security for various communication ports is determined by the security profiles a platform administrator creates.

Security profiles define the security characteristics of a client/server session. Assign a security profile to a listener, which is configured as a port that accepts client connection requests of various protocols. Unwired Server uses multiple listeners. Clients that support the same characteristics can communicate to Unwired Server via the same port defined in the listener.

Before you create a security profile in Sybase Control Center as part of the Unwired Server configuration, you must always create key pair of public and private keys and install that key pair in the keystore used by Unwired Server. You can then complete the security profile setup by specifying the key file, alias (also called a label) and keystore password.

Once you have created the security profile, configure the listener to use a security profile communicating over the encrypted protocol.

Configuring Security Profiles

Configure security profiles to secure communication between Unwired Server administration, DCNs, and client applications.

Prerequisites

Before creating a security profile, ensure that you possess digital certificates that have been verified and signed by third-party trusted authorities, as well as import required certificates in to the Unwired Server keystore. See the next topic, *Security Key and Certificate Basics*.

Task

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

3. In the right administration pane, select the **General** tab.
4. From the menu bar, select **SSL Configuration**.
5. Create a new Security Profile:
   a) Name the security profile.
   b) Enter the certificate alias for the profile (defined in the server keystore).
   c) Select the Authentication option.
6. Click **Save**.
7. In the server restart dialog, click **OK**.
8. Restart the server for these changes to take effect.

Next
Use the profile to encrypt administration and DCN ports.

Security Key and Certificate Basics
You can configure Unwired Server to accept connections over the secure protocols IIOPS and HTTPS by managing certificates and keys in the Unwired Platform security repository.

.

Table 7. Key and Certificate Management in Unwired Platform

Item	Details
Private key and certificate types	Supported types are “pkcs12” and “jks.”
Utility to maintain keystores and truststores	The SUN Java keytool utility, or the Sybase **createkey** utility.
Certificate assignment to a listener	In Sybase Control Center using a security profile you create. Ensure that the certificate has been imported into the Unwired Server keystore.
Location of the keystore and truststore	In `<UnwiredPlatform_Install-Dir>\Servers\UnwiredServer\Repository\Security`, use `keystore.jks` for the keystore and `truststore.jks` for the truststore. The keystore holds all server-side certificates (private keys); the truststore holds the trusted certificate authority (CA) root certificates.

Encrypting Unwired Server Administration Connections

Enable Unwired Server administration encryption to protect administration metadata that is transferred between Unwired Servers and Sybase Control Center.

Setting up administration encryption requires that you create certificates and import them into their respective keystores, and then use IIOPS to connect to a specified port. Sybase recommends that you create a security profile to associate a certificate with this port. You can get a certificate from a certificate authority (CA), or you can use the **createcert** utility to generate a self-signed certificate.

Configuring Unwired Server Administration Certificates

By default, Unwired Server includes two security profiles: "default" and "default_mutual." The "default" security profile uses the "sample1" certificate and the "default_mutual" security profile uses the "sample2" certificate. The certificate used by any other user-created security profiles is specified during security profile creation.

Prerequisites

- Determine whether the SSL security profile for the secure management (IIOPS) port requires "default" or "default_mutual" authentication. If the security profile requires "default" authentication, the Sybase Control Center truststore should contain the Unwired Server certificate. If it requires "default_mutual" authentication, the Sybase Control Center and Unwired Server truststores should each contain a copy of the other's certificate.
- By default, the keystore and truststore passwords are both "changeit". In a production deployment of Unwired Server, use the **keytool** utility to set new passwords for both of these files.

These steps describe the basics of exporting and import a certificate. Use the same steps to import your certificate into Unwired Server and Sybase Control Center keystore.

Task

1. Set up the certificates:
   a) Add %JAVA_HOME% to your system path.
   b) At a command prompt, change to `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Security`.
   c) Export the "default" security profile certificate `sample1.crt` from the Unwired Server keystore by running:

      ```
      keytool -keystore keystore.jks -storepass changeit -alias
      sample1 -exportcert -file sample1.crt
      ```

      See *System Administration > System Reference > Command Line Utilities > Certificate and Key Management Utilities > Key Tool (keytool) Utility*.

You can also create and import new certificates. For more information about how to generate a new keystore and truststore, see *http://docs.sun.com/app/docs/doc/819-3658/ablrb?a=view*.

2. Configure Sybase Control Center:
   a) Open `<UnwiredPlatform_InstallDir>\SCC-XX\services\Messaging\lib\eas\lib\Repository\Server\EmbeddedJMS\Instance\com\sybase\djc\server\ApplicationServer\EmbeddedJMS.properties`.
   b) Insert these values to the keystore and truststore files :

```
keyStore=<filePath>/<keyStoreName>.jks
keyStorePassword=<password>
trustStore=<filePath>/<trustStoreName>.jks
trustStorePassword=<password>
```

   Or, you can copy the Unwired Server keystore and truststore files and use them for Sybase Control Center instead.
   c) Import `sample1.crt` into the Sybase Control Center keystore by running:

```
keytool -keystore keystore.jks -storepass changeit -alias
sample1 -importcert -file sample1.crt
```

3. If you are running multiple Unwired Servers as part of a clustered environment, ensure these changes are repeated on all nodes. If certificates are issued to a specific host, you must generate a new certificate for each node.

Next

Enable and configure the administration port to use the security profile in Sybase Control Center.

See also

- *Key Creation (createkey) Utility* on page 303

Enabling and Configuring Administration Encryption for Unwired Server

Enable encryption to securely transfer data between the Unwired Server administration listener and Sybase Control Center.

You can create or change a security profile that saves SSL setup data for a particular server instance. Using the security profile, you associate a specific key with the encrypted port.

1. In the left navigation pane, expand the **Servers** folder and select a server.
2. Select **Server Configuration**.
3. In the right administration pane, click **General**.
4. Optional. If you want to create a new security profile, select **SSL Configuration**.
5. In the **Configure security profile table**:

a) Enter a name for the security profile.
b) Enter a certificate alias. This is the logical name for the certificate stored in the keystore.
c) Select an authentication level:
   - **intl** – select this type of authentication if only the server must provide a certificate to be accepted or rejected by the client.
   - **intl_mutual** – select this type of authentication if the client is also required to authenticate using a certificate; both the client and server will provide a digital certificate to be accepted or rejected by the other.

6. Enable the use of IIOPS in the **Communication Ports** subtab by selecting **Secure Management Port** (port 2001). See *Sybase Control Center > Configure > Configuring Unwired Platform > Unwired Server > Server Properties > General Server Ports > Configuring Communication Port Properties*.
7. Select the correct security profile name that provides the details for locating the correct certificates.
8. Save the changes and restart the server.
9. Repeat these steps on all remaining servers in the cluster.

Connecting to the Unwired Server Secure Administration Port

Manage the Unwired Platform cluster resource through Sybase Control Center using a secure administration port.

Prerequisites

Enable SSL encryption on each Unwired Server and restart all nodes in the cluster.

Task

1. In the Sybase Control Center Perspective Resources window, right-click the cluster resource and click **Properties**.
2. Change the port number to match the secure management (IIOPS) port. The default is 2001. This enables a secure connection when you log in to the cluster.
3. In the **Use Secure Connection** field, enter `yes`.
4. Click **OK**.
5. Right-click the cluster resource and select **Manage**.

Encrypting Relay Server Connections

Sybase recommends that you encrypt communication to and from a relay server in a production environment.

1. *Reviewing Relay Server Encryption Considerations*

Understand all relay server production considerations.

2. *Creating a Certificate Authority*

   Create RSA certificates to be used as the certificate authority (CA). Unwired Platform does not support ECC certificates.

3. *Generating a Certificate Request*

   Configure the Web server to use a certificate authority (CA) certificate that you create or another certificate signed by a CA.

4. *Signing the Server Certificate*

   Sign the server certificate request with the createcert utility.

5. *Installing the CA Certificate and Configuring the Server*

   Once you have created and reviewed the certificate, you can safely install it on the server.

Reviewing Relay Server Encryption Considerations

Understand all relay server production considerations.

- Each Relay Server in the production environment requires individual server certificates.
- If using a load balancer the load balancer must also be configured to use SSL. (See your load balancer documentation for details on what may be required.) In this case, Relay Server still needs to be configured to use SSL encryption, otherwise up and down channels between the RSOE and the servers are not encrypted over HTTPS.

See also

- *Creating a Certificate Authority* on page 93

Creating a Certificate Authority

Create RSA certificates to be used as the certificate authority (CA). Unwired Platform does not support ECC certificates.

1. At a command prompt, change to `<UnwiredPlatform_InstallDir>UnwiredPlatform\Servers\SQLAnywhere11\BIN32`.
2. Run:

   createcert -s

   The **-s** option generates a root certificate.

3. When prompted, enter `1024` as the **RSA key length**. For all remaining prompts, enter appropriate values for your deployment; for example:

```
<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers
\SQLAnywhere11\BIN32>createcert
SQL Anywhere X.509 Certificate Generator Version 11.0.1.2405
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
```

```
State/Province: CA
Locality: Dublin
Organization: MyCompany
Organizational Unit: PTO
Common Name: MyCompany
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
```

Note: You must use only an RSA Transport Layer Security (TLS) certificate, and not a certificate generated from ECC TLS or from a third-party source such as **openssl**, or by using a **createcert**- produced certificate from another Sybase product installation.

4. Record your private-key file path and password values, and the certificate and identity file paths. You will need these values again when fulfilling certificate requests for the Web server.

See also

- *Certificate Creation (createcert) Utility* on page 300
- *Reviewing Relay Server Encryption Considerations* on page 93
- *Generating a Certificate Request* on page 94

Generating a Certificate Request

Configure the Web server to use a certificate authority (CA) certificate that you create or another certificate signed by a CA.

You can use either the server's administration tool or the **createcert** utility to generate the certificate request.

1. To create a signed certificate request on the Web server:
   a. Open your server's administration tool. For example, on IIS, open IIS Manager.
   b. Use the administration tool to create a new certificate request, and save the text file.
2. To create a certificate request with **createcert**:

a. At a command prompt, change to `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\SQLAnywhere11\BIN32`.

b. Run:

createcert -r

c. Follow the prompts to complete the required information. For example:

```
<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers
\SQLAnywhere11\BIN32>createcert -r
SQL Anywhere X.509 Certificate Generator Version 11.0.1.2405
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
State/Province: CA
Locality: Dublin
Organization: MyCompany, Inc
Organizational Unit: Engineering
Common Name: MyCompany
Enter file path to save request: certreq.txt
Enter file path to save private key: myclient_private.crt
Enter password to protect private key: sybase
```

See *System Administration > System Reference > Command Line Utilities > Certificate and Key Management Utilities > Create Certificate (createcert) Utility*.

Use the generated CA certificate file to sign the server certificate, or do so using the root CA certificate, as described in *System Administration guide > Security Administration > Security Layers > Transport Security Setup > Encrypting Relay Server Connections > Signing the Certificate*.

See also

- *Certificate Creation (createcert) Utility* on page 300
- *Creating a Certificate Authority* on page 93
- *Signing the Server Certificate* on page 95

Signing the Server Certificate

Sign the server certificate request with the createcert utility.

Prerequisites

Ensure you have created a certificate request either by using the Web server's administration tool or by running `createcert -r`.

Task

1. At a command prompt, change to `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\SQLAnywhere11\BIN32`.
2. Run:

createcert and use the -s option to point to the certificate request file you created previously.

For example:

```
createcert -s certreq.txt
```

3. When prompted, enter the root certificate and its private key file and password created earlier to sign this certificate request. For all remaining prompts, enter appropriate values for your deployment; for example:

```
<UnwiredPlatform-InstallDir>\UnwiredPlatform\Servers
\SQLAnywhere11\BIN32>createcert -s certreq.txt
SQL Anywhere X.509 Certificate Generator Version 11.0.1.2405
Enter file path of signer's certificate: rsa_root.crt
Enter file path of signer's private key: rsa_private.crt
Enter password for signer's private key: sybase
Serial number [generate GUID]: <enter>
Generated serial number: ca8295aa17ca41a9a4bd0a3f613c0886
Certificate valid for how many years (1-100): 20
Certificate Authority (Y/N) [N]: N
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 [3,4,5]: <enter>
Enter file path to save certificate: myserver.crt
```

The `myserver.crt` file is installed on the server. Record the file information for future reference.

See also

- *Generating a Certificate Request* on page 94
- *Installing the CA Certificate and Configuring the Server* on page 97

Viewing Certificates

To ensure the certificate was created correctly, run the viewcert utility.

1. At a command prompt, change to *<UnwiredPlatform_InstallDir>*`\`*UnwiredPlatform*`\Servers\SQLAnywhere11\BIN32`.
2. Run:

   viewcert `<myCertFile>.crt`

   In this example, **viewcert** indicates that `rsa_root.crt` is a certificate authority (CA) certificate:

```
<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers
\SQLAnywhere11\BIN32>viewcert rsa_root.crt
```

```
SQL Anywhere X.509 Certificate Viewer Version 11.0.1.2405

X.509 Certificate
-----------------
Common Name:             MyCompany
Country Code:            US
State/Province:          CA
Locality:                Dublin
Organization:            MyCompany,Inc
Organizational Unit:     PTO
Issuer:                  MyCompany
Serial Number:           c24f9fa714db4f8084d9235dffdf9dcc
Issued:                  May 26, 2010   9:32:00
Expires:                 May 27, 2030   9:32:00
Signature Algorithm:     RSA, SHA1
Key Type:                RSA
Key Size:                1024 bits
Basic Constraints:       Is a certificate authority, path length
limit: 10
Key Usage:               Certificate Signing
```

In this example, **viewcert** indicates that `myserver.crt` is a server certificate:

```
<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers
\SQLAnywhere11\BIN32>viewcert myserver.crt
SQL Anywhere X.509 Certificate Viewer Version 11.0.1.2405

X.509 Certificate
-----------------
Common Name:             MyCompany
Country Code:            US
State/Province:          CA
Locality:                Dublin
Organization:            MyCompany, Inc
Organizational Unit:     Engineering
Issuer:                  MyCompany
Serial Number:           ca8295aa17ca41a9a4bd0a3f613c0886
Issued:                  May 26, 2010   9:34:00
Expires:                 May 27, 2030   9:34:00
Signature Algorithm:     RSA, SHA1
Key Type:                RSA
Key Size:                1024 bits
Basic Constraints:       Is not a certificate authority
Key Usage:               Key Encipherment, Data Encipherment, Key
Agreement
```

3. Review the values to ensure that the certificate information is correct.

Installing the CA Certificate and Configuring the Server

Once you have created and reviewed the certificate, you can safely install it on the server.

The specific details required for each server type varies. See the documentation that is available with either IIS or Apache. However, we can generalize the steps as follows.

Regardless of the server type, the general steps are:

1. Open your server's administration tool.
   For example, on IIS, open IIS Manager.
2. Process the pending request to install the certificate using the appropriate mechanism. Browse to the certificate and select the correct `*.CRT` file.
   For example, if you followed the steps described earlier, you are installing `rsa_root.crt`, and `myserver.crt`.
3. Set the SSL port to 443.
4. If you require client certificates, ensure you appropriately configure your Web server. In most cases, client certificates are not required, especially if you are using an LDAP provider.
5. Verify the HTTPS connection in the server by opening a browser and typing `https://<servername>` in the Address bar.
   An Under Construction message indicates that the certificate has been installed correctly.

See also

- *Signing the Server Certificate* on page 95

Encrypting Replication-Based Synchronization Connections

In a production environment for replication-based synchronization (RBS), you must set up both Unwired Server and device clients to use an encrypted port, and manage certificates on both the server and client.

These steps are required to secure communication over the RBS port in an environment where Unwired Server is running in the DMZ or as part of a development environment without a relay server:

1. *Creating Self-Signed Certificates*

   Use createcert to create a self-signed certificate that encrypts replication-based synchronization (RBS) connections. In a production environment, request the certificate from a trusted certificate authority (CA).
2. *Configuring Unwired Server to Use HTTPS for Replication Sync*

   Enable SSL encryption by configuring the replication-based synchronization HTTPS port.
3. *Verifying Device Client HTTPS Setup*

   Complete the encryption of the replication-based synchronization, by verifying the setup of device clients. Typically, device clients are set up by development teams.

Creating Self-Signed Certificates

Use createcert to create a self-signed certificate that encrypts replication-based synchronization (RBS) connections. In a production environment, request the certificate from a trusted certificate authority (CA).

1. At a command prompt, change to `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\SQLAnywhere11\BIN32`.
2. Run:

   createcert

3. When prompted, enter `1024` as the **RSA key length**. For all remaining prompts, enter appropriate values for your deployment; for example:

```
<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers
\SQLAnywhere11\BIN32>createcert
SQL Anywhere X.509 Certificate Generator Version 11.0.1.2405
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
State/Province: CA
Locality: Dublin
Organization: MyCompany
Organizational Unit: PTO
Common Name: MyCompany
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
```

   See *System Administration > System Reference > Command Line Utilities > Certificate and Key Management Utilities > Certificate Creation (createcert) Utility*.

Use the `myserver_identity.crt` file when you configure RBS encryption in Sybase Control Center and use the `mypublic_cert.crt` file when you configure the RBS application that connects to that port. If you are running the RBS server behind a relay server that already uses encryption, these follow-up steps are optional.

Next
Ensure you store your key and identity files in a safe place.

See also
- *Configuring Unwired Server to Use HTTPS for Replication Sync* on page 100

Configuring Unwired Server to Use HTTPS for Replication Sync
Enable SSL encryption by configuring the replication-based synchronization HTTPS port.

1. In the left navigation pane of Sybase Control Center for Unwired Platform, expand the **Servers** node and click the server name.
2. Click **Server Configuration**.
3. In the right administration pane, on the **Replication** tab, click **Synchronization Listener**.
4. Select `Secure synchronization port` as the protocol used for synchronization and configure the certificate properties. In the optional properties section, specify the `myserver_identity.crt` certificate file using the fully qualified path to the file, along with the password you entered during certificate creation.

   See *Sybase Control Center > Configure > Configuring Unwired Platform > Unwired Server Configuration > Configuring Unwired Server Properties > Replication > Configuring Replication-Based Synchronization Properties.*.

See also
- *Creating Self-Signed Certificates* on page 99
- *Verifying Device Client HTTPS Setup* on page 100

Verifying Device Client HTTPS Setup
Complete the encryption of the replication-based synchronization, by verifying the setup of device clients. Typically, device clients are set up by development teams.

See the applicable device platform developer reference for details about how to establish a secure connection with or without Relay Server.

1. Ensure the application code has been modified to use the HTTPS protocol, port, and stream parameters (with or without a relay server as is appropriate for your environment).
   - If you use relay server, and followed the previous steps in the *Encrypting Replication-Based Synchronization Connections* task workflow, the application developer use these settings in the code to connect to the relay server's secure port. For example:
     - Port – 443.
     - Protocol – HTTPS (this is equivalent to the MobiLink stream type).
     - Stream parameter – `"url_suffix=/ias_relay_server/client/rs_client.dll/[SUP_FARM_ID];tls_type=RSA;trusted_certificates=rsa_r`

```
oot.crt;identity=id_client.pem;identity_password=pwd;
"
```

Note: The `identity=id_client.pem;identity_password=pwd` segments of the stream parameter are only required if you use a relay server HTTPS port (requires client certificate mutual authentication). This configuration allows the relay server to block denial-of-service attacks at the periphery of you network, should you require that degree of security.

- If you use Unwired Server in the DMZ or in a development environment, and followed the previous steps in the *Encrypting Replication-Based Synchronization Connections* task workflow, the application developer uses these settings in the code to connect the secure port:
  - Unwired Server Port – 2481.
  - Protocol – HTTPS (this is equivalent to the MobiLink Stream Type).
  - Stream Parameter – `trusted_certificates=mypublic_cert.crt`

2. Make the `rsa_root.crt` and `id_client.pem` available for the application on the device. They can be included in the application or deployed separately.

See also

- *Configuring Unwired Server to Use HTTPS for Replication Sync* on page 100

Encrypting Messaging-Based Synchronization Connections

Messaging-based synchronization requires no manual setup.

Even though the messages are exchanged using HTTP, messages are fully secured because messaging-based synchronization uses a combination of symmetric (1024-bit) and asymmetric key encryption (128-bit AES) to ensure data privacy.

Encryption keys are based and rotated on a per message basis, not a per session basis. A session is the communication required to transmit item such as MBO data, which has multiple individual messages, each encrypted with its own encryption key. This method ensures key rotation and avoids key stagnation challenges.

Encrypting DCN Connections

Developers are responsible for the construction of data change notification (DCN) requests transmitted from the enterprise information server (EIS) . However, an administrator must enable and configure the HTTPS port for DCN connections, when the DCN goes directly to Unwired Server and does not use a relay server.

If you are using a relay server, then you only need to enable HTTPS on the RSOE using the relayserver.protocol property in the `relayserver.properties` file. See *System Administration > Environment Setup > Relay Server Setup* and *System Administration > System Reference > Configuration Files > Relay Server Configuration Files > Relay Server Properties (relayserver.properties) Configuration File*.

Otherwise, for direct DCN connections to Unwired Server, administrators configure a security profile.

1. In Sybase Control Center for Unwired Platform, expand the server node in the corresponding cluster.
2. In the left navigation pane, click **Server Configuration**.
3. In the right administration pane, click **General**.
4. In the **Communication Ports** subtab, expand **Hide secure data change notification ports**, and:
   a) Optionally enter a new port number. The default is 8001.
   b) Enable the desired secure ports.

      If the port value in the **Status** column is **disabled**, click the field and select **Enabled**.
   c) Enter a maximum thread count for each port.

      The value must be a positive integer. The default is 30.
   d) Select an appropriate security profile for the port by clicking the corresponding profile name in the **SSL Security Profile** column, and selecting a profile.
5. If you plan to use a new certificate for this port, create a new certificate, optionally sign it, import it into the Unwired Server keystore (see *Configuring Unwired Server Administration Certificates*, then use the new 104 Sybase Unwired Platform certificate when creating the security profile. Then, assign the security profile to the appropriate DCN ports.
6. If necessary, increase the **Max Thread** default property value.
7. Save the changes and restart the server.
8. Repeat on all particpating servers in the cluster

See also

- *Configuring Relay Server Timeouts for DCNs* on page 74
- *Data Change Notifications* on page 163

Encrypting Afaria Server Connections for Devices

Enable Afaria device client encryption to protect data that can be transferred between Afaria Servers and remote clients. The security settings are defined, modified and enforced on the client devices using the standard Afaria Security Manager channel. If channel encryption is insufficient, Afaria also allows administrators to encrypt the entire disk as well.

Enabling and Configuring Administration Encryption for Afaria

Configure Afaria to communicate with clients over SSL.

Prerequisites

Ensure you have Afaria installed and configured properly.

Task

1. Open Afaria Administrator.
2. Configure the correct combination of client communication properties. For details, see:
   - *Afaria Platform Reference Guide > Server Configuration*
   - *Afaria Administrator online help>Properties>Client Communication*

Generating an Afaria Certificate

Create a new certificate to encrypt only Afaria communications.

1. Run the Certificate Creation wizard.
   a) In Afaria Administrator, click **Server List** from the menu at the top of the right pane.
   b) Under the Roles column, click **Administrator**.
   c) On the **Home/Server Status** page, click **Server configuration** from the menu at the top of the right pane.
   d) In the left pane, click **Client communication**.
   e) In the Certificate Creation area, click **Generate certificate request**.
2. To create your certificate request and private key, follow the instructions in the wizard and click **Finish**.
3. Submit the request to a certificate authority (CA) to receive your certificate.

For more information on the Certificate Creation wizard, click the Help link at the top of the Afaria Administrator window and open the *Server Configuration* documentation.

See also

- *Certificate Creation (createcert) Utility* on page 300

Sharing an Unwired Server Certificate with Afaria Server

Use the same certificate for both Afaria and Unwired Server SSL encryption.

You can share a certificate only when the Afaria Server and Unwired Server are installed on the same host, unless you create wildcard certificates (for example, where you specify the domain name of the certificate to be *.<domain>). Wildcard certificates may not always be accepted by clients.

1. Use the **createcert** utility to generate the Unwired Server certificate request.

   Note: If Afaria requires a different format, use a public-domain utility that converts certificates among different formats.
2. Submit the request to a certificate authority (CA) to receive your certificate.

See also

- *Certificate Creation (createcert) Utility* on page 300

User Security Setup

User security ensures that enterprise resources are properly secured when back-end corporate data is mobilized to a device client. Login controls check individual identities, and restrict access to resources on the Unwired Platform-enabled system.

The login security and access control features in Unwired Server include several elements that secure data access from device clients. A security provider is an important part of user security. You can configure one or more security providers within a single security configuration. Creating multiple security configurations administrators and domain administrators to scope security stringency according to need for domains, packages, or operations as required.

A provider uses its own repository, which is a source of information about the users, security roles, security policies, and credentials.

Authentication

A security provider verifies the identities of application users and administrators who request access.

The two types of authentication (users versus administrators) are configured differently:

- For device users – authenticated with custom Unwired Server security configurations created by the platform administrator in Sybase Control Center. A default provider (OpenDS LDAP) is installed with Developer Edition and is suitable for development purposes. However, in a production environment, an enterprise-level security provider must be configured. See *Security for Device Users* later in this chapter.
- For administrators – authenticated with the "admin" security configuration that already exists in Sybase Control Center by default. See *Security for Administration Users* later in this chapter.

See also

- *LDAP Configuration Properties* on page 264
- *NTProxy Configuration Properties* on page 271
- *Domino Configuration Properties* on page 273
- *Remedy Configuration Properties* on page 274
- *RADIUS Configuration Properties* on page 275

Authentication Cache Timeouts

Unwired Server can cache authentication credentials for users.

Add a CacheTimeout property to the instance properties of the SecurityDomain definition of the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer \Repository\Instance\com\sybase\djc\security\SecurityDomain`

`\default.properties` file, to allow a user to connect to Unwired Server with cached credentials — even if the configured authentication provider is unavailable. That means, if a user logs in successfully, then during the configured CacheTimeout duration, the same user can re-authenticate using the same credentials without validating them against a security repository. This properties file should only be changed by platform administrators, and applies only for the packages which use 'admin' security configuration. For all other packages that use other security configurations, add/change the property in the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer \Repository \Instance\com\sybase\djc\security\<security configuration>.properties` instead.

Note: This value on affects authentication results; authorization results are not cached with this property. Also, if the user provides a user name or password that is different from the ones cached, Unwired Server then delegates the authentication request to the security repository.

If you want to restrict this type of access, ensure you set a low value for this property or disable cache timeouts by setting the value to 0.

Authorization

An access policy rule describes which user identity or role is authorized to access a specific resource under named or configured conditions. Unwired Server can be configured to include an authorization provider to authorize device users to access sensitive enterprise data. Sybase Control Center can also authorize administrators to access the complete set of Unwired Platform administration functionality, but domain administrators to only domain specific parts.

Unwired Server authorizes devices users with an authentication provider defined in a new security configuration configured by the platform administrator in Sybase Control Center. By contrast, Sybase Control Center authorizes administrators with the "admin" security configuration in the "default" domain.

Security providers only act as policy enforcers, and therefore do not allow administrators or domain administrators to directly:

- Define, modify, grant, revoke, or delete policies in the enterprise security repository. Only the security administrator has that privilege.
- Manage roles in the enterprise security repository.

  If you require roles, the physical role must exist in the repository. Administrators then later map the logical role used in the mobile business object to this physical role equivalent.

Instead, you must use the third-party administration tools available exclusively for that purpose.

Audit

You can audit security activity by configuring an auditing provider for Unwired Server and Sybase Control Center. The audit provider tracks events such as logins, logouts, server start

operations, and so on, including all actions performed by a specific user or with a particular role.

Sybase suggests that administrators audit two types of user activity: successful use of sensitive information and failed attempts at accessing unauthorized information.

Note: On a properly configured and tuned server with adequate capacity, enabling auditing should have minimal impact on the overall system performance.

Security Configurations

A security configuration determines the scope of user identity, authentication and authorization checks, and can be assigned to one or more domains for both messaging and replication-based applications. For example, user "John" may be authenticated one way in one security configuration, but authenticated differently another security configuration, even when "John" represent the same human user.

A security configuration aggregates various security mechanisms for protecting Unwired Platform resources under a specific name, which administrators can then assign to domains. Each security configuration consists of:

- A set of configured security providers
- Security roles
- Role mappings (set at the domain and package level)

A user entry must be stored in the security repository used by the configured security provider to access any resources (that is, either a Sybase Control Center administration feature or an application package to access a data set from a back-end data source). When a user attempts to access a particular resource, Unwired Server tries to authenticate and authorize the user by checking the:

- security policy on the requested resource
- role memberships

Creating and Applying a Security Configuration

Administering security configurations requires an SUP administrator role. Domain administrators cannot modify the assigned security configuration; instead, they can map roles for the configuration at the domain and package level.

Security configuration administration follows this sequence:

1. A platform administrator:
   a) Creates a security configuration.
   b) Creates and configures the providers used by the security configuration. You must configure at least one authentication provider, however, you can use multiple providers of different types.

c) Assigns the security configuration to the domain. You must assign at least one security configuration.
d) Assigns domain administration privileges to users.

2. Once assigned a domain and a security configuration, the domain administrator can set the security configurations for the packages in the domain. If the packages use any logical roles, the domain administrator can also perform role mapping between the logical roles and physical roles of the package security provider.

Stacking Multiple Security Providers

Stack multiple providers to provide a security solution that meets more complex security requirements.

Each provider has a controlFlag attribute that controls overall behavior when you enable two or more providers.

1. Use Sybase Control Center to create a security configuration and add multiple providers as required for authentication, authorization, attribution, and audit. For details, see *Sybase Control Center > Configure > Configure Unwired Platform > Security*.
2. For each provider:
   a) Select the provider name.
   b) Click **Properties**.
   c) Configure the controlFlag property with one of the available values: required, requisite, sufficient, optional.

      See *controlFlag Attribute Values* for descriptions of each available value.
   d) Configure any other common security properties as required.
3. Click **Save**.
4. Select the **General** tab, and click **Apply**.

Security Provider Issues

If you experience problems with security configurations or the authentication or authorization providers in these configurations, you need to check the Unwired Server logs for issues.

If you are finding that no errors are being reported, despite failures that may occur while authenticating or authorizing users, you may need to increase the severity level of your logs. To increase severity levels, see *Sybase Control Center Online Help > Configure > Configuring Unwired Platform > Unwired Server > Server Logs Configuring Server Log Settings*.

See also

- *System Logs* on page 222

Built-in Security Providers

Unwired Server supports a variety of built-in security providers. Administrators define one or more security providers when they create a security configuration using Sybase Control Center.

No Security Provider

A NoSec provider offers pass-through security for Unwired Server, which is intended for use in development environments or for deployments that require no security control.

If you use NoSec providers, all login attempts succeed, no matter what values are used for the user name and password. Additionally, all role and control checks based on attributes also succeed.

Sybase has used these classes to implement the NoSec provider:

- **NoSecLoginModule** – provides pass-through authentication services.
- **NoSecAttributer** – provides pass-through attribution services.
- **NoSecAuthorizer** – provides pass-through authorization services.

LDAP Security Provider

The LDAP security provider includes authentication, attribution, and authorization providers.

You can configure these providers:

- The **LDAPLoginModule** provides authentication services. Through appropriate configuration, you can enable certificate authentication in LDAPLoginModule.
- Optional. The **LDAPAuthorizer** or **RoleCheckAuthorizer** provide authorization services for LDAPLoginModule. LDAPLoginModule works with either authorizer. In most production deployments, you configure your own authorizer. However, if you are authenticating against a service other than LDAP, but want to perform authorization against LDAP, you can use the LDAPAuthorizer.
- Optional. The **LDAPAttributer** provides attribution services.

You need not configure all LDAP providers. You can also implement some LDAP providers with providers of other types.

See also

- *LDAP Configuration Properties* on page 264

Active Directory Considerations

If you are using Active Directory either as a security provider, ensure you understand the implications of using this LDAP directory in a production environment of Unwired Platform.

Consider these design implications when you extend Active Directory security to Unwired Platform production environments:

- Shared identities among Sybase components – if you are using Active Directory for all authentication requests (that is, administration logins to access Sybase Control Center and application logins to access data), you must set up both Sybase Control Center and Unwired Server to use this Active Directory installation. This allows users to have a shared user identity in both components. However, the user identity must be already configured both as an Unwired Platform user as well as an administrator.
- LDAP data structure and administrative control – Unwired Platform requires that roles be added to the repository in specific locations, and therefore, that security configurations also correctly point to the location of the roles. Otherwise, authentication may be problematic. Coordinate the implementation of these roles with the LDAP administrator.

See also

- *LDAP Configuration Properties* on page 264

Multiple LDAP Trees for Authentication and Authorization

Use the Unwired Platform administration perspective to configure the LDAP authentication and authorization security providers, which are used locate LDAP user information when organizational user groups exist within multiple LDAP trees.

You can use these approaches to accommodate an LDAP tree structure that cannot be directly accessed using one search base:

- Create an LDAP authentication module for each level in the hierarchy – during the authentication process, Unwired Platform tries to authenticate against every login module in the ordered list until authentication succeeds or until it reaches the end of the list. Depending on the number of login modules you configure, this approach may have some performance issues.
- Use different scopes for performing user searches – specify the root node of a particular LDAP tree, by entering `AuthenticationSearchBase="dc=sybase, dc=com"` and `set Scope=subtree`. Unwired Platform performs an LDAP query against the entire subtree for authentication and authorization information. Depending on the number of subtrees within the LDAP tree structure, this approach can have performance implications.
- Implement a proprietary login module – create a custom logic and search mechanism by implementing the login module callback interface. This approach involves Java coding to a set of public interfaces within Unwired Platform.

NTProxy Security Provider

NTProxy (sometimes known as native Windows login) integrates with existing Windows login security. Users can authenticate with their native Windows user name and password, which gives them access to roles that are based on their existing Windows memberships.

The NTProxy provider fulfills authentication services only with classes in `csi-nativeos.jar`; role-based access control, and attribution are not directly supported. Groups are also not supported in NTProxy. Instead, group memberships are transformed into a

role of the same name and can be mapped in Sybase Control Center. See *Sybase Control Center > Configure > Configuring Unwired Platform > Role Mapping Configuration*.

See also

- *NTProxy Configuration Properties* on page 271
- *Setting Up the NTProxy Provider* on page 115

RADIUS Security Provider

The RADIUS security provider implements token-driven authentication against an existing RADIUS server.

The RADIUS provider fulfills authentication services only with the `com.sybase.security.radius.RadiusLoginModule` class; authorization and attribution services are not supported. If you need to configure these providers, you can use the LDAP providers with the RADIUS authentication provider.

RADIUS is not recommended for device user authentication: the application would require a different password each time it connects (generated by a token card), which would then require frequent user input to update that token password. Therefore RADIUS authentication should be limited to administration user authentication at most.

See also

- *RADIUS Configuration Properties* on page 275

Domino Security Provider

The Domino security provider performs only authentication and attribution. It uses the HTTP/DIIOP authentication mechanism supported by Domino server.

The Domino provider fulfills authentication services only with the **DominoLoginModule** class; authorization, and attribution is not directly supported because Domino's login provider offers only username and password authentication.

If you require authorization services, you can use one of the Domino providers along with other provider types. For example, if you require authorization services, you can use **DominoLoginModule** with **RoleCheckAuthorizer**.

See also

- *Domino Configuration Properties* on page 273

Remedy Security Provider

The Remedy security provider implements authentication and authorization against a Remedy AR system.

The Remedy security provider consists of:

- `com.sybase.security.remedy.RemedyLoginModule` provides authentication services against the Remedy server.
- `com.sybase.security.core.RoleCheckAuthorizer` provides authorization services using a core class from the security framework.

See also

- *Remedy Configuration Properties* on page 274

Roles and Mappings

Role mapping occurs when an administrator maps logical roles to physical roles using Sybase Control Center as part of a security configuration or a deployment package. The physical roles are the roles and groups in the underlying security repository. The mapped role determines the security role requirement for a user at runtime to access a resource that is using the security configuration on which the mapping is defined.

In Unwired Platform, the mapped role determines what security roles apply to users when they attempt to perform an operation from the mobile application (device users) or Sybase Control Center (administrators).

Role mappings are defined as part of a security configuration that you can assign to a particular domain. Administrators can assign the same security configuration to multiple domains; ensure that these mappings are suitable for all domains to which the security configuration is assigned. Consider an example where security configuration is shared between domainA and domainB.

1. The platform administrator (the administrator assigned the SUP administration role) creates a security configuration called AllDomains.
2. The platform administrator assigns the AllDomain to the domain, and maps the EmpRole role to SalesGroupRole in the security repository used by that configuration.

This change that is specific to just domainA is also implemented in domainB even though the domain administrator of domainB did not explicitly make, or require, the change. But the role mapping is propagated to domainB as well. To avoid this, the Unwired Platform administrator may want to create multiple security configurations so that underlying mechanisms can stay the same, but specific role mappings can be made for each.

For device user security, there is an increased flexibility for packages as they are deployed. If a security configuration is inappropriate, or if a role is not mapped at all that is used by the package, the platform or domain administrator can override or extend the role mappings defined for the security configuration. Package-level role mappings always take precedence in such a scenario.

See also

- *Mapping Roles for a Domain* on page 145
- *Mapping Roles for a Package* on page 152

Physical Roles

Physical roles are named references to roles or groups that an administrator has defined on a back-end enterprise security provider. Mapping a logical role to a physical role allows authorization control in the Unwired Server. Replicate these names exactly, so that logical roles can be mapped correctly to the server role.

Logical Roles

Logical roles are defined by package developers, and allow developers to define identities that each application uses to indicate access rights to its different objects. Logical roles may or may not replicate physical names already defined for your security provider.

Logical roles allow secured access to Unwired Platform resources for several users at once, and you can define them for one or more packages in a domain. If a developer has created a logical role for a package, you may need to manually map it to one or more physical roles.

In the absence of explicit mapping, the default role mapping is set to AUTO, which is equivalent of logical role mapping to a physical role of the same name, in the underlying provider of that security configuration.

Mapping State Reference

The mapping state determines the authorization behavior for a logical name instance.

State	Description
AUTO	Map the logical role to a physical role of the same name. Both the logical role and the physical role must match, otherwise, authorization fails.
NONE	Disable the logical role, which means that the logical role is not authorized. This mapping state prohibits anyone from accessing the resource (MBO or Operation). Use this option after carefully considering potential consequences.
MAPPED	A state that is applied after you have actively mapped the logical role to one or more physical roles. Click the cell adjacent to the logical role name and scroll to the bottom of the list to see the list of mapped physical roles.

Dynamically Mapping Physical Roles to Logical Roles

Map roles at either a domain or package-level, depending on the scope requirements of a particular binding. If you use a particular role mapping for a package and a different role mapping at the domain-level, the package mapping overrides the domain-level mapping.

In Sybase Control Center for Unwired Platform, determine where the role mapping needs to be applied:

- For domain-level mappings, configure role mappings as part of the security configuration for a domain.
- For package-level mappings, configure role mappings when you deploy a package to Unwired Server, or at the package-level after deployment.

Security for Administration Users

Sybase Control Center requires its own set of security providers, which are distinct from those you must set up for Unwired Platform security. Typically, the only providers an administrator would configure for Sybase Control Center would be authentication and authorization.

1. *Configuring a Security Provider for Sybase Control Center*

   Once you have added required users to the repository used for Sybase Control Center authentication, you can use that directory to authenticate administration login requests.
2. *Setting Up Provider Roles in Sybase Control Center*

   Configure the security provider administration and user roles and passwords required for Sybase Control Center administrator login.
3. *Setting Up the Admin Security Configuration*

   Configure the admin security configuration to authenticate all administrator users. In the Developer Edition, the admin security configuration points to the OpenDS LDAP server and the default role mapping. To secure your Unwired Server infrastructure in the Deployment Edition, you must use Sybase Control Center to set the admin security configuration to point to your security repository server and configure the appropriate role mapping.
4. *Mapping and Assigning Unwired Platform Administration Roles*

   Unwired Platform uses two logical roles: SUP Administrator and SUP Domain Administrator. You must use Sybase Control Center to map these logical roles to the appropriate physical roles or groups in the underlying security provider.

Configuring a Security Provider for Sybase Control Center

Once you have added required users to the repository used for Sybase Control Center authentication, you can use that directory to authenticate administration login requests.

Prerequisites

Backup the `<SCC_HOME>\conf\csi.properties`.

To use Windows Native authentication, ensure you setup the Windows NTProxy login provider.

Task

1. Exit Sybase Control Center.
2. From Windows Services panel, stop the Sybase Unified Agent Service.

3. From a text editor, open `<SCC_HOME>\conf\csi.properties`.
4. Define a module in this file, similar to the LDAP sample below. This example specifies that Active Directory is used as the LDAP server for Sybase Control Center authentication requests.

   Each line of the LDAP server module of the properties file must begin with "CSI.loginModule." followed by a module number. The module number in this sample is 5. The module number must be unique in the properties file, and you must use the same number in every line of the module.

```
=====================================================
## LDAP login module for SCC
CSI.loginModule.
5.options.AuthenticationSearchBase=ou=users,dc=example,dc=com
CSI.loginModule.5.options.BindDN=cn=Directory Manager
CSI.loginModule.5.options.BindPassword=secret
CSI.loginModule.5.options.DefaultSearchBase=dc=example,dc=com
CSI.loginModule.5.options.ProviderURL=ldap://localhost:10389
CSI.loginModule.
5.options.RoleSearchBase=ou=groups,dc=example,dc=com
CSI.loginModule.5.options.ServerType=AD
CSI.loginModule.5.options.moduleName=SUP LDAP Login Module
CSI.loginModule.
5.provider=com.sybase.ua.services.security.ldap.LDAPWithRoleLogin
Module
CSI.loginModule.5.controlFlag=sufficient
=====================================================
```

   Note: Change the values for only lines shown in bold.

   For a complete list of available LDAP properties you can configure, see *System Administration > System Reference > Security Provider Configuration Properties > LDAP Configuration Properties*.
5. For some internal communication, you must include the Anonymous Login Module in the `csi.properties` file:

```
# Anonymous Login Module
CSI.loginModule.
0.provider=com.sybase.ua.services.security.anonymous.AnonymousLog
inModule
CSI.loginModule.0.controlFlag=sufficient
CSI.loginModule.0.options.moduleName=Anonymous Login Module
CSI.loginModule.0.options.roles=uaAnonymous
```

   Adding this anonymous login module does not relax or allow anonymous access to the Sybase Control Center. Authentication and authorization checks are still enforced..
6. Save the file.
7. If your LDAP server's SSL certificate is signed by a nonstandard certificate authority (for example, if it is self-signed), use the **keytool** utility to configure JVM to trust the certificate. Execute a command similar to this:

```
keytool -import -keystore <SUP_installdir>\shared\JRE-<version>
\bin\keytool\lib\security\cacerts -file
<your cert file and path> -alias ldapcert -storepass changeit
```

8. Restart Sybase Unified Agent.
9. Open Sybase Control Center and log in.

See also

- *Setting Up Provider Roles in Sybase Control Center* on page 116

Setting Up the NTProxy Provider

To use a Windows NTProxy login provider, set up the Windows account as well as an `admin` security configuration in Sybase Control Center.

This security provider is typically used only to configure administrator security in Sybase Control Center.

1. Add Unwired Platform groups and users to Windows by running the **Control Panel** and opening **User Accounts**.
   - Create a SUP Administrator and a SUP Domain Administrator group.
   - Add administrator users and passwords to that group. For example, the default user and password is supAdmin and s3pAdmin respectively.
2. Configure NTProxy for Sybase Control Center, by opening `SCC_HOME\conf\csi.properties`, and adding an NTProxy login module. For example:

   ```
   CSI.loginModule.
   5.provider=com.sybase.ua.services.security.os.NTProxyLoginModule
   CSI.loginModule.5.controlFlag=sufficient
   CSI.loginModule.5.options.moduleName=NT Proxy Login Module
   CSI.loginModule.5.options.debug=true
   ```

3. Manually map roles for Sybase Control Center, by opening `SCC_HOME\conf\roles-map.xml`, and editing the `NTProxy Login Module` entity to add SUP Administrator and SUP Domain Administrator group mappings.

   The "name" property must match the "moduleName" property value in csi.properties.

   Note: The value in "modRole" are Windows OS groups designated for Unwired Platform administrator and Domain administrator.

   This example uses "SUP Administrator" as the Windows OS group for platform administrator and "SUP Domain Administrator" for domain administrator rights. Make sure to create those groups or use your group name, and assign users membership to respective group as needed:

   ```
   <module name="NT Proxy Login Module">
         <role-mapping modRole="Administrators"
   uafRole="uaAnonymous,uaAgentAdmin" />
         <role-mapping modRole="Administrators"
   uafRole="uaAnonymous,uaOSAdmin" />
   ```

```
     <role-mapping modRole="Users" uafRole="uaAnonymous,uaUser" /
>
      <role-mapping modRole="Guests"
uafRole="uaAnonymous,uaGuest" />
      <role-mapping modRole="sybase"
uafRole="uaAnonymous,uaPluginAdmin,sccUserRole" />
      <role-mapping modRole="SUP Administrator"
uafRole="uaAnonymous,uaAgentAdmin,uaPluginAdmin,sccAdminRole,sccU
serRole" />
                        <role-mapping modRole="SUP Domain
Administrator"
uafRole="uaAnonymous,uaAgentAdmin,uaPluginAdmin,sccUserRole" />
</module>
```

You can also map roles in SCC. See *Sybase Control Center > Configure > Authorization > Assigning a Role to a Login or a Group*.

4. Configure the NTProxy provider for the "admin" security configuration in SCC.
   a) Log in as a SUP administrator.
      Supply the domain as well as the user name, for example: `userName@domainName`
   b) In the left navigation tree, click **Security** > **admin** to modify the existing security configuration used for SCC.
   c) In the right administration pane, select the **Authentication** tab and click **New**.
   d) Select NTProxy as the login module, and ensure that the **Default Authentication Server** property value is the domain name of the SCC server host computer.
   e) Configure other properties as required and click **OK**.
   f) Select the Unwired Server name, and click **Apply** to save the changes to the `admin` security configuration.

The next time you log in to SCC, use your Windows user account name and password to authenticate as an administrator or domain administrator. A domain name is still required. For example: instead of supAdmin/s3pAdmin, use **userName@domainName**/WindowsPwd.

See also

- *NTProxy Security Provider* on page 109
- *NTProxy Configuration Properties* on page 271

Setting Up Provider Roles in Sybase Control Center

Configure the security provider administration and user roles and passwords required for Sybase Control Center administrator login.

1. Exit Sybase Control Center.
2. From Windows Services panel, stop the Sybase Unified Agent Service.
3. Ensure that the roles defined in the LDAP repository match the roles defined in the `<SCC_HOME>\conf\roles-map.xml` file.

   By default, the role mapping file contains these roles in the LDAP Login Module definition:

```
 <module name="SUP LDAP Login Module">
      <role-mapping modRole="SUP Administrator"
uafRole="uaAnonymous,uaAgentAdmin,uaPluginAdmin,sccAdminRole,sccU
serRole" />
      <role-mapping modRole="SUP Domain Administrator"
uafRole="uaAnonymous,uaAgentAdmin,uaPluginAdmin,sccAdminRole,sccU
serRole" />
    </module>
```

You can change "SUP Administrator" and "SUP Domain Administrator" to reflect the role names that you want to use for administrator and domain administrator authentication.

4. To add role mapping for the Anonymous Login Module to the uaAnonymous role:
   a) Add the uaAnonymous role to the `<uaf-roles>` section of the `roles-map.xml` file:

      ```
      <role name="uaAnonymous" description="Anonymous role" />
      ```

   b) Add the role mapping in the `<security-modules>` section of the `roles-map.xml` file:

      ```
      <module name=''Anonymous Login Module''>
                        <role-mapping modRole=''uaAnonymous''
      uafRole=''uaAnonymous'' />
          </module>
      ```

5. Restart Sybase Unified Agent.
6. Open Sybase Control Center and log in.

See also

- *Configuring a Security Provider for Sybase Control Center* on page 113
- *Setting Up the Admin Security Configuration* on page 117

Setting Up the Admin Security Configuration

Configure the `admin` security configuration to authenticate all administrator users. In the Developer Edition, the `admin` security configuration points to the OpenDS LDAP server and the default role mapping. To secure your Unwired Server infrastructure in the Deployment Edition, you must use Sybase Control Center to set the `admin` security configuration to point to your security repository server and configure the appropriate role mapping.

1. In the left navigation pane, expand the **Security** folder.
2. Select **Admin**.
3. In the right administration pane, configure authentication and authorization by clicking the corresponding tab, and configuring properties of the providers you add as required.

   By default, authentication and authorization uses No Security. You can customize the security configuration by removing this provider and other types of providers.
4. In each tab, remove the No Security modules:
   a) Click the box adjacent to the No Security module name.

b) Click **Delete**.

5. Configure the same LDAP provider that is used to authenticate and authorize Sybase Control Center administrative users. You can use a single provider can be used for both Sybase Control Center and Unwired Platform administrators.
   a) Click **New**.
   b) Select the provider.
   c) Set property values that are suitable for your security requirements.
   d) Click **OK**, to save the configuration locally.
6. Click **Save** and restart Unwired Server to commit the changes.

See also

- *Setting Up Provider Roles in Sybase Control Center* on page 116
- *Mapping and Assigning Unwired Platform Administration Roles* on page 118

Mapping and Assigning Unwired Platform Administration Roles

Unwired Platform uses two logical roles: SUP Administrator and SUP Domain Administrator. You must use Sybase Control Center to map these logical roles to the appropriate physical roles or groups in the underlying security provider.

The mapping of these logical roles to one or more physical roles in the underlying security provider decides whether a platform or domain administrator has access privileges. The administration logical to physical role mapping is done in 'default' domain for the 'admin' security configuration at the domain-level.

1. In the left navigation pane, expand **Domains**.
2. Expand the **Default** domain.
3. Open the Security folder and click the **Admin** security configuration.
4. Map roles to the security provider groups or roles:
   - If administration roles are already part of the security provider repository, select **AUTO**. The roles in the repository must exactly match to the administration role names for Unwired Platform.
   - To map the administration role to one that exists in the repository, but that differs from the role names for Unwired Platform, click the list adjacent to the logical role and choose **Map Role**. This command displays the Role Mappings dialog, which allows you to manually set the logical and physical role mappings. The dialog shows the name of the logical role you are mapping in the text area of the dialog. Once saved, the state automatically changes to MAPPED.

   See *Sybase Control Center > Configure > Configure Unwired Platform > Domains > Configuring Domain Security*.
5. Assign domain administration access to users in Sybase Control Center:

a) Register the user by clicking the **Security** node and selecting the **Domain Administrators** tab, then clicking **New**.
b) Assign the required domain administrator physical role to the user in the underlying security provider repository for **admin** security configuration.

See *Sybase Control Center > Configure > Configure Unwired Platform > Domains > Administrators*.

6. Restart Unwired Server.

See also

- *Setting Up the Admin Security Configuration* on page 117

Security for Device Users

Unwired Server requires administrators to configure security provider for device users. The production environment for your device applications may require you to create multiple security configurations of different types of providers.

Use Sybase Control Center to configure security providers for device users security.

For example, a company sales employee needs to look up a client's phone number in a phone book device application. This authentication sequence allows the sales employee to access data from the phone book application:

1. The employee tries to open the application, which prompts for a user name and password, which is local to the device, and not explicitly tied to a corporate security account.
2. The first time the application is opened, the employee must synchronize the customer MBO to access the client phone number.
3. Unwired Server gets an authentication request.
4. Unwired Server sends the request to the authentication provider that processes the login credentials.
5. The provider checks the user name and password against information stored in authentication repository, in this case, an LDAP directory server on the corporate LAN.
6. The directory server evaluates the access policy to see if the authenticated user has permission to access this client's contact information.
7. If the login request is valid, the user is authenticated. Because the employee has the correct access privileges, Unwired Server is notified and the resource request is fulfilled.

   Note: The granularity of access control checks is at the MBO-class or MBO-operation level. Therefore, if the user has access to one customer record, he or she can access all customer records.
8. If the login request is invalid, an error is generated and authentication fails.

Data Security Setup

Administrators can implement data security in two ways: changing passwords required to access the component databases used in Unwired Platform, and encrypting data in these repositories.

1. *Protecting System Data Access*

   Data stored in cluster and consolidated database can contain varying degrees of sensitive data; Sybase recommends that you immediately change the default DBA password to one that is more secure.
2. *Data Encryption Implementation*

   Encrypt the data used by Unwired Platform to add an additional layer of platform security.

Protecting System Data Access

Data stored in cluster and consolidated database can contain varying degrees of sensitive data; Sybase recommends that you immediately change the default DBA password to one that is more secure.

1. *Changing the Consolidated Database DBA Password*

   Run the Interactive SQL utility (dbisql) to change the DBA password.
2. *Registering Password Changes Among Components*

   Update the modified password in server configuration files.
3. *Verifying the DSN Entries*

   After you have changed the DBA passwords, verify that the data sources still work.

See also

- *Data Encryption Implementation* on page 122

Changing the Consolidated Database DBA Password

Run the Interactive SQL utility (**dbisql**) to change the DBA password.

Prerequisites

Ensure runtime servers are stopped. In Windows, select **Start** > **Programs** > **Sybase** > **Unwired Platform<version>** > **Stop Unwired Platform Services**.

Task

1. Change to `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\bin32`, and run:

   ```
   dbisql
   ```

2. Locate the database you are changing the password for:
   a) Select **ODBC data source name**.
   b) Click **Browse**.
   c) Select the database. You may need to select **Show all data sources** to see all names used by Unwired Platform.

      For the consolidated database, choose **default-CDB** if you are using the default SQL Anywhere consolidated database.
   d) Click **OK**.
3. Enter:

   ```
   grant connect to dba identified by <NewPwd>
   ```
4. Execute the command by clicking the right arrow.
5. Exit **dbisql**.
6. Repeat these steps for the cluster database. If you are using the default DSN names, the cluster DSN is clusterdb_<clustername>.

See also

- *Registering Password Changes Among Components* on page 121

DBA Passwords

By default, the Unwired Platform creates many component databases, and Unwired Server accesses these databases with the DBA user identity.

For the default consolidated database of SQL Anywhere, a single user ID of DBA is created, and the password is initially sql (passwords are case-sensitive). The DBA user ID automatically gives Unwired Server the DBA authority within these component database, which enables Unwired Server to perform any activity in the database. Unwired Server can create required tables, change table structures, and so on. For this reason, Sybase strongly recommends you change these default passwords to increase the level of protection to Unwired Platform cache and metadata.

Registering Password Changes Among Components

Update the modified password in server configuration files.

1. Open `<UnwiredPlatform_InstallDir>\Sybase\UnwiredPlatform\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer\sup.properties`.
2. Modify the cdb.password and cldb.password properties to set a new password.

   Passwords are entered in clear text password. After you have completed the next step, all passwords are encrypted and the sup.properties file is resaved with the new encrypted values.
3. Run `<UnwiredPlatform_InstallDir>\Sybase\UnwiredPlatform\Servers\UnwiredServer\bin\configure-mms.bat <clustername>`.

In a single server setup, <clustername> is host name of this computer. In a cluster setup, <clustername> is the hostname of first node installed in the cluster.

4. Repeat this step on all server nodes in cluster.

Next

Validate that the data source has been updated with the correct password.

See also

- *Changing the Consolidated Database DBA Password* on page 120
- *Verifying the DSN Entries* on page 122

Verifying the DSN Entries

After you have changed the DBA passwords, verify that the data sources still work.

1. Select **Control Panel > Administrative Tools > Data Sources (ODBC)**.
2. Select the **System DSN** tab.
3. Double-click **default-cdb**, then click **Test Connection**.
   Look for a `"Connection Successful"` message.
4. Return to the **System DSN** tab.
5. Double-click **clusterdb_<*cluster name*>**, then click **Test Connection**.
   Look for a `"Connection Successful"` message.

See also

- *Registering Password Changes Among Components* on page 121

Data Encryption Implementation

Encrypt the data used by Unwired Platform to add an additional layer of platform security.

Some Unwired Platform components do not support encryption. Review this table to see which components can enable this security feature.

Component	Encryption available?	Implementation notes
Various embedded databases	No	Not applicable

Component	Encryption available?	Implementation notes
Device data	Yes, but optional	Use the Data Security Manager with Afaria and encrypt either some of the data or the entire disk. Encrypted data is always decrypted before it is packaged and transported. Therefore, Sybase recommends that if security is a concern, that you encrypt the communication channel used.
Device client database	Yes, but optional	A developer can enable database encryption by using either an API or a database property. A developer can encrypt PIM data or entity application data—in other words, personal and enterprise data that is cached persistently on the device, and is normally accessed by the user via a corresponding Unwired Platform mobile workflow or mobile business object. Encrypted data is always decrypted before it is packaged and transported. Therefore, Sybase recommends that if security is a concern, that you encrypt the communication channel used.

See also

- *Protecting System Data Access* on page 120
- *Chapter 9, Administration Client API* on page 251

Encrypting Data on the Windows Device Client

You can use Afaria to enable data encryption. Afaria data encryption uses a component called Data Security Manager, which is available only with the correct license of Sybase Unwired Platform.

Each Afaria client supports a different level of data encryption.

1. Contact Sybase about obtaining the correct license for device data encryption.
2. With the correct license in place, choose the client type and level of encryption desired.

   Only some device clients are shared by Unwired Server and Afaria:

Encryption level	Device client	Implemented by
Full disk	Windows 32	Administrator

Encryption level	Device client	Implemented by
PIM data	Windows 32, Windows Mobile, and Symbian	Administrator selects th PIM data that Afaria can encrypt calendar, e-mail, and so on.
Path/folder	Windows 32, Windows Mobile/Windows Mobile Professional, and Symbian	Administrator defines the encryption target. However, a user can copy files to the encrypted area of a disk.
Path/folder	Windows Mobile Professional	User defines encryption target.

You can find complete information about configuring data encryption in Afaria documentation. See *Afaria Reference/Components > Data Security Manager for Handheld Clients > Encryption Options* guide for details. For information about which subtypes of Windows Mobile are supported by the Data Security Manager License, see the system requirements documented in the *Afaria Release Notes*.

CHAPTER 7 System Administration

System administration includes all administration activities performed by an administrator that support effective day-to-day operations of Unwired Platform components in a production environment.

System administration for Unwired Platform may use only a single Unwired Platform administrator (that is, the user granted the SUP Administrator role). However, larger deployments may use multiple team members, distributed roles that include both Unwired Platform administrators and domain administrators, as well as other supporting information technology (IT) members like customer support technicians, or database administrators.

From a product life cycle perspective, administration include installation, setup, maintenance, and administration actions, that use various tools:

Action	Task	Frequency	Tool
Component installation	Installing to deploy components to create the development/test or production environment.	Infrequent	Unwired Platform installer
Postinstallation environment setup	Configuring servers, setting up security.	Infrequent	Sybase Control Center
Environment maintenance	Changing configuration or security as system evolves.	Infrequent	Sybase Control Center
MBO package administration	Deploying packages, defining server connections, setting role mapping, managing subscriptions, and managing cache groups, cache schedule, and synchronization groups.	Routine	Sybase Control Center
Mobile environment administration	Registering users and allowing them to subscribe to data, as well activating, configuring, and provisioning devices.	Routine	Sybase Control Center Afaria Administrator Afaria OTA Manager
Monitoring the system	Inspecting various activities to monitor performance and identify bottlenecks.	Frequent	Sybase Control Center

Server Environment Administration

Server environment administration includes all the activities required to set up and configure your Unwired Platform server environment.

Administration of the server environment includes:

- Managing Unwired Server clusters. The membership (adding or removing a node), naming, and unlinking of a cluster's relay server is all handled during the installation or manual configuration process.

  Versioning of the cluster occurs automatically and cannot be directly controlled by the administrator. If you have a cluster-affecting change on the primary, the change is shared with the rest of the cluster.

  Note: Before you set up the cluster, plan the details accordingly. For planning information, see *System Design* chapter.
- Configuring the server environment from Sybase Control Center.
- Enabling and designing domains for multiple tenants in your system.
- Managing and monitoring your consolidated database. Decide whether to use a new default database or an existing one, and configure the environment to use the corresponding database. You can do this with the installer or with Sybase Control Center postinstallation (for existing databases only).

Cluster Administration Overview

The goal of cluster administration is to ensure that clusters and servers work smoothly, and scale over time. Cluster administration is mostly a nonroutine administration task.

By default, the Unwired Platform is installed as a one-node cluster, which may be sufficient for development or test environments. However, production deployments of Unwired Platform of 25 users or more are likely to require multiple nodes. See *Sybase Unwired Platform System Administration > Systems Design*.

Table 8. 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

Task	Frequency	Accomplished by
Setting cluster properties, including consolidated database settings, monitoring database setup, and so on	Once, or as cluster changes require	Manual configuration using files and `.BAT` scripts.
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. Consult with your database administrator. See *Sybase Unwired Platform System Administration > Database Management*.
Reviewing licensing information, including total licensed devices and currently used licenses count	Occasional, or as device user registration and deregistration occurs	Sybase Control Center

See also

- *Server Administration Overview* on page 127
- *Setting Up Runtime Server Tier Nodes* on page 55

Server Administration Overview

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

Table 9. Server administration tasks

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

Task	Frequency	Accomplished by
Configuring the server to: • Set the replication and messaging synchronization ports, as well as communication ports for administration and DCN • Create security profiles for secure administration communication • Set up secure synchronization • View consolidated database properties • Configure replication and messaging push notifications • Set transport level security properties within a security profile • Tune server performance	Postinstallation configuration with infrequent tuning as required	Sybase Control Center for Unwired Platform. See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server*.
Setting server log file settings and subsystem log levels	Once, unless log data requirements change	Sybase Control Center for Unwired Platform. See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Logs*.

See also

- *Data Mobility* on page 19
- *Cluster Administration Overview* on page 126
- *Setting Up Runtime Server Tier Nodes* on page 55
- *Installing Third-party Software* on page 56

Configuring Listener Behavior

By default, Unwired Server listeners are configured to listen on the first IP address resolved for your host name.

To configure Unwired Server to listen on all network interfaces, edit each listener's properties file.

1. Stop Unwired Server.
2. On the system where Sybase Unwired Platform is installed, navigate to the `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\Repository\Instance\com\sybase\djc\server\SocketListener` directory.
3. Use a text editor to open the `<hostname>_iiop1.properties` file.

4. Locate the line beginning with `host=`:

```
#Instance Properties
#Mon Feb 22 10:34:27 PST 2010
maxThreads=200
port=2000
host=<your_host_name>
ant.project=default-socket-listeners
useSocketChannel=false
protocol=iiop
```

5. Replace your host name with `0.0.0.0`:

```
#Instance Properties
#Mon Feb 22 10:34:27 PST 2010
maxThreads=200
port=2000
host=0.0.0.0
ant.project=default-socket-listeners
useSocketChannel=false
protocol=iiop
```

6. Save and close the properties file.
7. Restart Unwired Server.

Viewing Consolidated Database Properties

Review the consolidated database (CDB) properties that allow Unwired Server to connect to the database.

You cannot use Sybase Control Center to configure CDB properties. If you are changing the CDB password, ensure you review the contents of *Sybase Unwired Platform System Administration Guide > Security Administration > Security Layers > Data Security Setup*.

1. In the left navigation pane, expand the **Servers** folder and select a server.
2. Select **Server Configuration**.
3. In the right administration pane, click the **Consolidated DB** tab.
4. Review these properties:
   - **Database Thread Count** – the number of worker threads used for the CDB. The default value is 20 threads. However, if you are experiencing performance issues, especially in a clustered environment, you may need to increase this value.

     For a SQL Anywhere CDB only, use this formula to estimate a new value:

     ```
     Value = Number of nodes in cluster * (sync threadcount +
     1) + Number of scheduled EIS fetches + 10
     ```

     For example, if you have been using the default synchronization thread count of 20, but have added three Unwired Servers to your cluster, adjust the CDB thread count to 78 or (3 * (20 +1) + 5 + 10). If you set the value for this thread count to 78 or higher, the value is accepted. However, if you set the value lower than 78, the request is ignored, unless you remove some servers from the cluster or reduce the synchronization thread counts.

- **Database Type** – the CDB type; Sybase_ASA for the default SQL Anywhere database.
- **Database DSN Name** – CDB DSN name descriptor.
- **Database Name** – database name descriptor.
- **Database Server Port** – the port over which CDB communication takes place. The default is 5200.
- **Database Password** – the database user password. In SQL Anywhere, the default is `sql`.
- **Database Server Host** – the name of the machine where the existing database server is running.
- **Database Server** – the name of the database server used to manage requests for the consolidated database. By default, the server is localhost. However, if the database is on another host or is part of a cluster, you may need to use another host name.
- **Database User** – the CDB user name. In SQL Anywhere, the default user name is `dba`.

5. (Optional) Expand the **Show Optional Properties** section and review values for these properties:
   - **Database Install Type** – the type of database installation; either default or custom.
   - **Database ASA Mode** – the database mode. The value of the first node of a cluster is "primary," the value of the second node is "arbiter," and the value of the third node is "mirror."
   - **Database User Options** – specify the CDB start-up user options.

Configuring Replication-Based Synchronization Properties

Configure the port to receive replication-based synchronization requests from client devices, and if you are using push synchronization, then also configure synchronization listener properties.

Prerequisites

Determine whether you require an encrypted (secured) or unencrypted synchronization stream. A secure synchronization stream uses SSL encryption; therefore, before setting up a secure configuration, ensure that you possess digital certificates verified and signed by third-party trusted authorities. The HTTPS protocol is slower than the HTTP protocol; use SSL only if you require HTTPS. See *System Administration > Security Administration > Security Layers > Transport Security Setup*.

Task

1. In the left navigation pane, expand the **Servers** folder and select the server you want to configure.
2. Select **Server Configuration**.

3. In the right administration pane, click the **Replication** tab.
4. If push synchronization is being added to replication-based synchronization application, select **Synchronization Listener** from the menu bar.
5. Select the protocol and port you require:
   - If you do not require SSL encryption, choose **Synchronization port**. Sybase recommends this option if you do not require a secure communication stream for synchronization. By default, the port for HTTP is 2480.
   - To encrypt the HTTP stream with SSL, choose **Secure Synchronization port**. By default, the port for HTTPS is 2481.
6. Configure these properties:
   - **Synchronization Cache Size** – sets the maximum cache size for the synchronization port. The default is 50MB.
   - **Thread Count** – sets the number of worker threads used for synchronization. The default is 5. If you experience performance issues, you may need to increase this value.
7. (Optional) Expand the optional properties section to configure these properties:
   - **Certificate Password** – identifies the password that Unwired Server uses to unlock the security certificate.
   - **Certificate** – identifies the location of the security certificate used to encrypt and decrypt data transferred using SSL.
   - **User Options** – sets the command line options for starting the synchronization server. These options are appended the next time the synchronization server starts. These are the available user options:

Option	Description
`@ [variable | filePath]`	Applies listener options from the specified environment variable or text file.
`-a <value>`	Specifies a single library option for a listening library.
`-d <filePath>`	Specifies a listening library.
`-e <deviceName>`	Specifies the device name.
`-f <string>`	Specifies extra information about the device.
`-gi <seconds>`	Specifies the IP tracker polling interval.
`-i <seconds>`	Specifies the polling interval for SMTP connections.
`-l <"keyword=value;...">`	Defines and creates a message handler.
`-m`	Turns on message logging.

Option	Description
`-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.
`-ot <filePath>`	Truncates a file, then logs output to that 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.
`-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 *Listener options for Windows* in the *SQL Anywhere 11.0.1* online help.

8. Click **Save**.

Configuring Messaging-Based Synchronization Properties

Configure one or more synchronization ports to receive service requests from devices.

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

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

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

Configuring System Performance Properties

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

1. In the left navigation pane, expand the **Servers** folder and select a server.
2. Select **Server Configuration**.
3. In the right administration pane, select the **General** tab.
4. From the menu bar, select **Performance Configuration**.
5. Configure these properties, as required:

   - Host Name – the name of the machine where Unwired Server is running.
   - Thread Stack Size – the JVM `-Xss` option.
   - Minimum Heap Size – the minimum size of the JVM memory allocation pool, in megabytes. Sybase recommends that this value not fall 500 megabytes for a 32-bit operating system, but 1 gigabyte is recommended. 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).
   - Maximum Heap Size – the maximum size of the JVM memory allocation pool, in megabytes. For a 32-bit operating system, Sybase recommends a 1.5 gigabyte maximum heap size value. For a 64-bit operating system, Sybase recommends 1 gigabyte for a normal configuration, but 4 gigabyte for a stress configuration (which can vary depending on what RAM is available).

   Note: Always leave 4 gigabytes for the running of the OS and other applications that may exist on the server.
6. (Optional) Expand the **Show optional properties** section and configure these properties, as required:

   - User Options – other JVM options. For example, you can enable JVM garbage collection logging by setting `-XX:+PrintGCDetails`.
   - Inbound Messaging Queue Count – the number of message queues used for incoming messages from the messaging-based synchronization application to the server. Sybase recommends a choose a value that represents at least 10% of active devices.

- Outbound Messaging Queue Count – the number of message queues used for outbound messages from the server to the messaging-based synchronization application. Sybase recommends a choose a value that represents at least 50% of active devices. However, if you are running 32-bit operating system, do not exceed a value of 100% of active devices.

7. Click **Save**.
8. If your server is installed as a Windows service:
   a) Stop Unwired Server.
   b) Open a command prompt.
   c) Run `sup-server-service.bat remove`.
   d) Run `sup-server-service.bat install auto`.
   e) Restart Unwired Server.

See also

- *Server Performance Tuning* on page 75

Applying Performance Tuning Changes if Unwired Server is a Service

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

If you installed Unwired Server as a Windows service, then follow these recommendations after changing the configuration of certain properties.

For the CDB pool size, no additional action is required.

1. For Java min/max heap size, and Java thread stack size properties:
   a) Change the property values in Sybase Control Center.
   b) Stop Unwired Server.
   c) Open a command prompt.
   d) Run `sup-server-service.bat remove`.
   e) Run `sup-server-service.bat install auto`.
   f) Restart Unwired Server.
2. For DCN listener thread count, restart the server.

See also

- *Server Performance Tuning* on page 75

SNMP Notifications

You can set up Sybase Unified Agent 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 Unified Agent, 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 Unified Agent 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 Unified Agent 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 Unified Agent service.

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.

Task

1. Stop Sybase Unified Agent.
2. Enable the SNMP plug-in to run when Sybase Control Center starts:
   a) Open `<UnwiredPlatform_InstallDir>\<SCC-XX>\plugins\com.sybase.supsnmpplugin_1.5.2\agent-plugin.xml`.
   b) Set `register-on-startup="true"`.

c) (Optional) Modify the value of the `sup.server.ping.schedule.interval` property to specify how often (in seconds) the SNMP plug-in pings Unwired Server to detect the server status. The default is 100.

3. (Optional) Change the SNMP notification target to a custom destination:
   a) Open `<UnwiredPlatform_InstallDir>\<SCC-XX>\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.

See also

- *Handling Transmitted SNMP Notifications* on page 136

Handling Transmitted SNMP Notifications

Configure an SNMP notification handling tool to process Unwired Platform SNMP notifications.

Prerequisites

Install an SNMP notification handling tool, such as HP OpenView.

Task

1. On the SNMP notification target host computer, launch an SNMP notification handling tool.
2. Using the third-party documentation, configure the SNMP notification handling tool to process Unwired Platform SNMP notifications. Configure the notification handler to listen for notifications on the port specified in the "snmp.notification.targets" property of the `<UnwiredPlatformInstall>\<SCC-XX>\services\Snmp\service-config.xml` file.

See also

- *Enabling SNMP Notifications for Unwired Platform* on page 135
- *Testing Notifications with SNMP Queries* on page 137

Testing Notifications with SNMP Queries
Test the configuration of the Unwired Platform SNMP plug-in for Sybase Control Center by using a management information base (MIB) browser tool to execute an SNMP query.

Prerequisites
Install a third-party MIB browser tool.

Task

1. Load `<UnwiredPlatformInstall>\<SCC-XX>\plugins\com.sybase.supsnmpplugin_1.5.2\SYBASE-SUP-MIB.txt` as a module into your MIB browser.
2. Configure the MIB browser settings to use an SNMPv3 information module with the parameters specified in the `<UnwiredPlatformInstall>\<SCC-XX>\services\Snmp\service-config.xml` file:

Property name	Description	Default value
snmp.transport.mappings	SNMP agent host/port	UDP (0.0.0.0/1498) The host "0.0.0.0" represents the local host. Changing this value to a different IP or host name causes the service to fail, since the SNMP service functions only on the local host. The default port number is 1498. You can set a different port for SNMP queries, however, you must ensure that it is not occupied by another SNMP agent.
snmp.usm.user	User name	snmpadmin
snmp.auth.passphrase	Auth password	Sybase4me

3. In the MIB browser, set the **Auth Protocol** to SHA and the **Security Level** to `Auth,NoPriv`.
4. In the object identifier tree of the MIB browser, navigate to `SYBASE-MIB\enterprises\sybase\sup\supObjects\supStatusTable\supStatusEntry` and select **get SNMP variable**.
   Unwired Server status information appears in the data console.

See also
- *Handling Transmitted SNMP Notifications* on page 136

Domain Administration Overview

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

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

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

- An Unwired Platform administrator adds and configures domains, creates security configurations for customer applications, and assigns those security configurations to the domain so they can be mapped to packages in the domain. You can use a dedicated security repository like LDAP or the tenant's own provider via a secure VPN-like connection. See *System Administration > Security Administration*.
- One or more domain administrators then perform domain-level actions within their assigned domains.

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

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

Table 10. Domain management tasks

All tasks are performed in Sybase Control Center.

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

Task	Frequency	Administrator
Monitor domain logs	Routine	Unwired Platform administrator and domain administrator
Deploy packages	Routine	Unwired Platform administrator and domain administrator
Manage server connections and templates	Infrequent, as required	Unwired Platform administrator and domain administrator
Manage subscriptions	Routine	Unwired Platform administrator and domain administrator
Review client log and MBO/ operation error history	Routine	Unwired Platform administrator and domain administrator

See also

- *Multitenant Environments* on page 58

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*

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

3. *Creating a Security Configuration for a Domain*

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

4. *Activating a Domain Administrator*

   A platform administrator must create and register domain administrators, before this individual can assign the domain administrator to 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.

Determining a Tenancy Strategy

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

Domains are primarly containers for packages for a specific group of users. 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.

See also

- *Creating and Enabling a New Domain* on page 141

Creating and Enabling a New Domain

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

See also

- *Determining a Tenancy Strategy* on page 140
- *Creating a Security Configuration for a Domain* on page 141

Creating a Security Configuration for a Domain

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

A security configuration determines the scope of data 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 *Sybase Control Center online help > Configure > Configuring Unwired Platform > Security Configurations*.

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.

See also

- *Creating and Enabling a New Domain* on page 141
- *Activating a Domain Administrator* on page 142

Activating a Domain Administrator

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

Prerequisites

Domain administrator physical roles should already be mapped to the default SUP Domain Administrator logical role per mapping in 'admin' security configuration in 'default' domain. However, this role name may differ if you choose to use a domain administration role from your company's repository.

Task

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

   See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Domains > Domain Administration > Registering a Domain Administrator User*.
2. Assign the login the required physical role in the security provider repository that authenticates and authorizes administrative logins for Sybase Control Center and Unwired Server. See *Sybase Control Center online help > Configure > Configure Sybase Control Center > Authorization*.

See also

- *Creating a Security Configuration for a Domain* on page 141
- *Assigning Domain Administrators to a Domain* on page 143

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. In the left navigation pane, expand the **Domains** folder, and select the domain for which to assign domain administration privileges.
2. Select the domain-level **Security** folder.
3. In the right administration pane, select the **Domain Administrators** tab, and click **Assign**.
4. Select one or more administrator users to assign to the domain by checking the box adjacent to the user name.
5. Click **OK**.
   A message appears above the right administration pane menu indicating the success or failure of the assignment. If successful, the new domain administrator appears in the list of users.

See also

- *Activating a Domain Administrator* on page 142

Managing and Maintaining Domains

Configure domain components, including datasource connections, logging, role mappings, and packages.

These tasks can be performed by both platform and domain administrators.

1. *Creating Data Source Connections*

   Use Sybase Control Center to configure the properties required to connect to datasources.
2. *Enabling and Configuring Domain Logging*

   Activate or deactivate domain logging in Sybase Control Center, and configure domain log autopurge settings for all nodes in a cluster. Domain logging collects data that pertains to the activities of all packages in a domain. You must have administrator privileges to configure domain logging.
3. *Mapping Roles for a Domain*

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

4. *Deploying an MBO Package to a Domain*

   Deploy a MBO package to Unwired Server using Sybase Control Center. Packages are initially created by developers, but are deployed and maintained on a production Unwired Server by administrators.

Creating Data Source Connections

Use Sybase Control Center to configure the properties required to connect to datasources.

A connection is required to send queries to mobile business objects and receive data. 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 *Sybase Control Center online help > Configure > Configuring Unwired Platform > Connections*.

Note: When creating connections, please ensure that the prerequisites for each connection type are installed on all nodes of the cluster. See *System Administration guide > Environment Setup > EIS Connections*.

1. Select the domain-level **Connections** node for the domain to configure.
2. Create a new connection or connection template by selecting the appropriate tab and clicking **New**.
3. 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.
4. 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.
5. 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

- *Enabling and Configuring Domain Logging* on page 144

Enabling and Configuring Domain Logging

Activate or deactivate domain logging in Sybase Control Center, and configure domain log autopurge settings for all nodes in a cluster. Domain logging collects data that pertains to the activities of all packages in a domain. You must have administrator privileges to configure domain logging.

First, domain-level logging must be enabled by a platform administrator. Domain-level logging controls whether package-level logging captures data. Then either the platform

administrator or the domain administrator can enable logging on a per-package basis from the Packages node of Sybase Control Center. See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Packages > Enabling Package Logging*.

1. In the left navigation pane, expand the **Domains** folder and select the domain for which to configure log settings.
2. Select **Log**.
3. In the right administration pane, select the **Settings** tab.
4. Select one of:
   - **Enable** – activate domain logging in Sybase Control Center.
   - **Disable** – turn off domain logging.
5. Set the autopurge threshold by entering the length of time (in days) to retain domain log data.
6. Click **Save**.

See also

- *Creating Data Source Connections* on page 144
- *Mapping Roles for a Domain* on page 145

Mapping Roles for a Domain

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

Prerequisites

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

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

Task

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

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

The states of AUTO or NONE require the least administration.

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

See also

- *Roles and Mappings* on page 111
- *Enabling and Configuring Domain Logging* on page 144
- *Deploying an MBO Package to a Domain* on page 146

Deploying an MBO Package to a Domain

Deploy a MBO package to Unwired Server using Sybase Control Center. Packages are initially created by developers, but are deployed and maintained on a production Unwired Server by administrators.

Platform or domain administrators can deploy replication and messaging packages as required.

In the left navigation pane, from the **Domain** node, launch the Deploy wizard. See *Sybase Control Center > Manage > Managing Unwired Platform > Routine Command and Control Actions > Deploy > Deploying a Replication or Messaging Package*.

See also

- *Mapping Roles for a Domain* on page 145

Deploying a Mobile Workflow Package to a Domain

Use Sybase Control Center to deploy a mobile workflow package to make it available on Unwired Server. Packages are initially created by developers, but are deployed and maintained on a production Unwired Server by administrators.

Platform administrators can deploy mobile workflow packages as required.

In the left navigation pane, from the **Workflows** node, launch the Deploy wizard for mobile workflow packages. See *Sybase Control Center > Manage > Managing Unwired Platform > Routine Command and Control Actions > Deploy > Deploying a mobile workflow Package*.

EIS Connection Management Overview

The goal of enterprise information system connection management is to ensure the connections to back-end repositories of data remain available to Unwired Server and deployed

packages that require those connections. Connections management is a non-routine administration task.

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

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

See also

- *EIS Connections* on page 80
- *EIS Data Source Connection Properties Reference* on page 277

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.

Note: When creating connections, ensure that the prerequisites for each connection type are installed on all nodes of the cluster. See System Administration guide > Environment Setup > EIS Connections.

See also

- *Data Source Connection Reference* on page 277

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.

Package Administration

Unwired Platform packages are administered differently depending on their type. However, all packages types must be deployed and configured.

- Replication-based synchronization mobile business object (MBO) packages use an application layer service that synchronously pulls data changes from the server. Because data updates occur in bursts because of periodic synchronization between the client and server, replication-based synchronization packages use the following package configuration features:
  - Subscription templates are created by the administrator and define how the device user is to be notified when cache data changes, depending on the subscription properties configured for that synchronization group in the package.
  - Cache groups define policies that determines the frequency and the level to which server data is refreshed and used by the packaged MBO. Create as many cache groups as required to meet the varying data refresh needs of the MBOs for a given mobile application.
  - Synchronization groups determine the frequency with which notifications are generated from Unwired Server to initiate MBO synchronization, and thereby access refreshed data. A synchronization group is a collection of MBOs that are synchronized together at regular intervals. Notifications are then delivered, depending on cache group schedule repeat, synchronization group change detection interval, and subscription notification threshold values.
- Message-based synchronization MBO packages create a device service that uses a dedicated channel that is always open to asynchronously push data changes, requests, and notifications between client and server. Because messaging-based MBO packages make use of both replicated and nonreplication (message) data to complete a mobile workflow process, they use these package configuration features:

    - Subscriptions are user-defined and specify the synchronization messages mobile device users receive and how they receive these messages. However, administrators can configure some properties to further refine and control the behavior.
    - Synchronization groups for messaging-based synchronization packages directly download the unit of changed data to the device, as opposed to sending a notification message. A synchronization group is a collection of MBOs that are synchronized together at regular intervals.
- Mobile workflow packages must be created with the Mobile Workflow Application Designer. This tool allows developer to design mobile workflow screens that can call on the create, update, and delete operations, as well as named queries, of a mobile business object.
    - E-mail settings allow the administrator to configure a listener to scan all incoming e-mail messages delivered to the particular inbox that the administrator indicates during configuration.
    - Matching rules are used by the e-mail listener to identify e-mail messages that match the rules specified by the administrator. When the e-mail messages match the rule, Unwired Server sends the e-mail message as a mobile workflow to the device that matches the rule.
    - Context variables customize how data is loaded into the Unwired Server cache. By determining the context variables you want to use, you can create a data set that is smaller, more focused, and therefore yields better performance.

Deployment

The last step of mobile business object (MBO) development is to deploy the MBO definitions to Unwired Server as a deployment unit generated from a design-time deployment package using Sybase Unwired WorkSpace.

When you deploy MBOs to the Unwired Server, you are deploying:

- MBO definitions including attributes, operations, connections, role mappings, schedule groups, cache groups as defined in the package.
- MBO custom code related to object filters, result-set filters, and result checkers.
- Appropriate generated server-side artifacts that support the interaction with the EIS back-ends and device application.
- Other functionality captured in the MBO model.

MBOs are deployed using a deployment wizard through which you can make the choices that are appropriate for application requirements. Developers use Unwired WorkSpace to deploy a package.

The production administrator can deploy from a wizard using the web-based management console, or from the command line. Deployment-time tasks include choosing:

- Target domain – logical container for packages.

- Security configuration – used for authentication and authorization of users accessing the package.
- Role-mappings – to map logical roles to the physical roles of the back-end repository.
- Server connections mapping – to bind MBOs design-time data sources to production data sources.

MBO Package Management Overview

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

Packages contain MBOs that are deployed to Unwired Server to facilitate access to back-end data and transactions from mobile devices. Package types include replication-based synchronization (RBS) packages, messaging-based synchronization (MBS) packages, and SAP DOE-C packages.

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

Table 11. MBO package management tasks

Task	Package type	Frequency	Accomplish by using
Deploy packages to a development or production Unwired Server	RBS and MBS	Once, unless a new version becomes available	Sybase Control Center for Unwired Platform with the domain-level Packages node
Control user access by assigning security configurations for each package, and mapping roles if fine-grained authorization is enforced through logical roles	RBS and MBS	Once, unless security requirements of the package change	Sybase Control Center for Unwired Platform with the domain-level Packages node
Set up the package cache interval and cache refresh schedule (for getting data updated on the Unwired Server from the data source)	RBS and MBS	Once, unless data refreshes need to be tuned	Sybase Control Center for Unwired Platform with the domain-level Packages node

Task	Package type	Frequency	Accomplish by using
Manage subscriptions (RBS, MBS, and DOE-C), synchronization groups (RBS and MBS), and device notifications (RBS) to customize how updated data in the cache is delivered to the device user	Varies	Periodic, as required	Sybase Control Center for Unwired Platform with the domain-level Packages node
Export or import an MBOpackage	RBS and MBS	On-demand, as required	Sybase Control Center for Unwired Platform with the domain-level Packages node
Review current/historical/performance metrics	All	Routine	Sybase Control Center for Unwired Platform with the Monitor node (available only to administrators)

See also

- *Occasionally Disconnected* on page 22
- *Replication-Based Synchronization* on page 23
- *Subscriptions* on page 174
- *Replication Devices* on page 173

Deploying and Managing MBO Packages

Use Sybase Control Center to deploy MBO packages created by developers to a production Unwired Server and manage package configuration.

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

Deploying an MBO Package to a Domain

Deploy a MBO package to Unwired Server using Sybase Control Center. Packages are initially created by developers, but are deployed and maintained on a production Unwired Server by administrators.

Platform or domain administrators can deploy replication and messaging packages as required.

In the left navigation pane, from the **Domain** node, launch the Deploy wizard. See *Sybase Control Center > Manage > Managing Unwired Platform > Routine Command and Control Actions > Deploy > Deploying a Replication or Messaging Package*.

Selecting a Security Configuration for a Package

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

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

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

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

Mapping Roles for a Package

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

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

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

See also

- *Roles and Mappings* on page 111

Enabling Package Logging

Information, errors and events can be recorded for packages.

Prerequisites

Package log data is sent to the domain log, only if you enable domain logging. Ensure you enable domain logging before enabling package logging.

Task

1. In the left navigation pane of Sybase Control Center, expand the **Packages** folder and select the package to configure.
2. In the right administration pane, click the **Settings** tab.
3. Enable package logging or synchronization tracing as required.
4. Click **Save**.

   View, search, and export package log data from the domain log. See *Sybase Control Center online help > Manage > Managing Unwired Platform > Routine System Maintenance Tasks > Checking the Domain Log*.

Mobile Workflow Package Administration Overview

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

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

Table 12. Mobile workflow package management

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

Enabling and Configuring the Notification Mailbox

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

The notification mailbox configuration uses a listener to scan all incoming e-mail messages delivered to the particular inbox specified during configuration. When the listener identifies

an e-mail message that matches the rules specified by the administrator, it sends the message as a mobile workflow to the device that matches the rule.

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

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

Deploying and Managing Mobile Workflow Packages

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

Multiple tasks are involved in workflow package deployment and management.

Configuring Mobile Workflow Package Properties

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

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

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

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

   See *Sybase Control Center > Configure > Configuring Unwired Platform > Workflows > Configuring a Mobile Workflow Package > Configuring Context Variables*.

Assigning and Unassigning Device Users

Assign mobile workflow packages to make them available to a registered device user. You can also unassign mobile workflow packages at any time.

1. In the left navigation pane, click **Workflows > MyWorkFlow**.
2. In the right administration pane, click the **Devices** tab.
3. Locate the device to assign a mobile workflow package to, then:
   a) Click **Assign Workflow**.
   b) List the users to assign the mobile workflow package to.

      By default, no users are listed in this window. Search for users by selecting the user property you want to search on, then selecting the string to match against. Click **Go** to display the users.
   c) Click **OK**.
4. To unassign a mobile workflow package, select the device user and click **Unassign Workflow**.

Deploying a Mobile Workflow Package to a Domain

Use Sybase Control Center to deploy a mobile workflow package to make it available on Unwired Server. Packages are initially created by developers, but are deployed and maintained on a production Unwired Server by administrators.

Platform administrators can deploy mobile workflow packages as required.

In the left navigation pane, from the **Workflows** node, launch the Deploy wizard for mobile workflow packages. See *Sybase Control Center > Manage > Managing Unwired Platform >*

Routine Command and Control Actions > Deploy > Deploying a mobile workflow Package.

Managing Deployed Package Subscriptions

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

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

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

Table 13. Subscription management tasks

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

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

Data Management Overview

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

Table 14. Data Management Tasks

Task	Frequency	Accomplish by using
Install a new or configure an existing consolidated database (CDB) for Unwired Platform.	Once	If you are installing a new CDB, no action is required. Otherwise, use either the installer, or Sybase Control Center (postinstallation) to configure the existing database to use as the CDB. For details about using Sybase Control Center, see *Sybase Control Center online help > Configure > Configure Unwired Platform > Unwired Server > Consolidated Database*.
Set up the cache group refresh schedule (to update data on the Unwired Server from the data source).	Once, unless data refreshes need to be tuned	Sybase Control Center
Create sync paradigms (for messaging-based sync packages), synchronization groups, and device notifications (for replication-based sync packages) to customize how updated data in the cache is delivered to the device user.	Periodic, as required	Sybase Control Center
Review current/historical/performance metrics	Routine	Sybase Control Center

See also

- *Data Mobility* on page 19
- *Database Clusters* on page 48
- *Backup and Recovery* on page 234

Data Mobility Configuration Dependencies

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

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

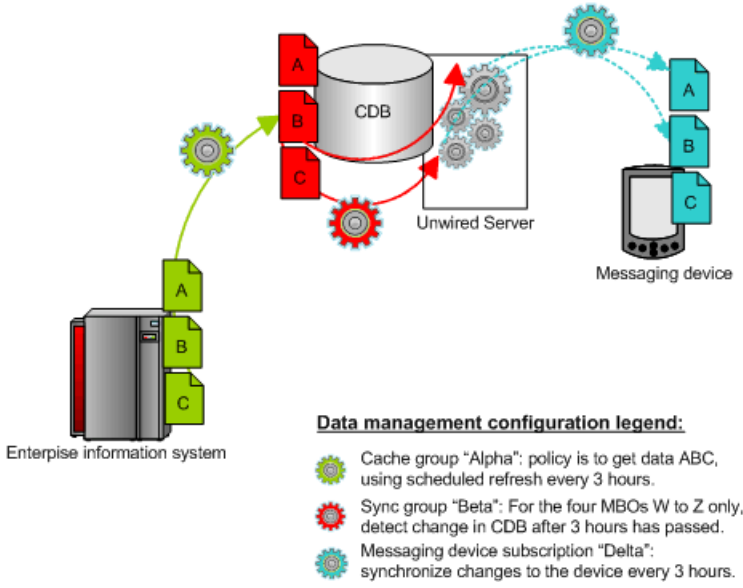
Setting	Sync type	Data updated	Determines	Dependencies
DCNs	Both	Server cache	The specific data units the EIS sends to Unwired Server cache. DCN updates operate outside of the cache group and cache refresh mechanisms.	None. No coordination required. However, data change notifications (DCNs) do trigger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

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

Message Data Flow and Dependencies

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

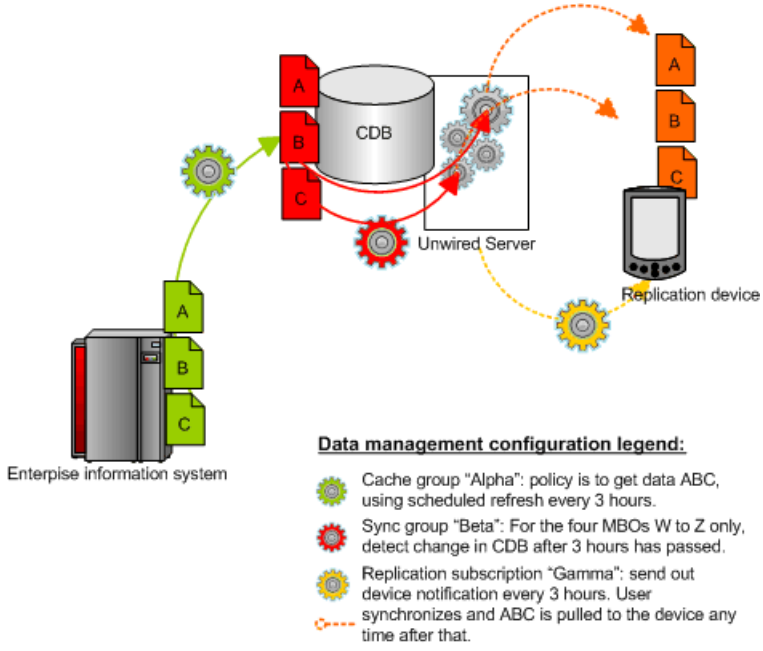
Cache refresh schedules, combined device settings and synchronization groups, work together to determine how data is disseminated to the front line from back-end EIS data sources. Coordinate these properties accordingly.



Replication Data Flow and Dependencies

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

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



Push Synchronization for Replication Packages

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

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

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

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

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

Enabling Push and Pull Notifications

Configure push and pull for replication-based synchronization notifications.

Prerequisites

Determine the type of push synchronization gateway you require.

This task sets the lightweight polling notification configuration for all clients. You can also configure it individually for each device.

Task

1. In the left navigation pane, expand the **Servers** folder and select the server to configure.
2. Select **Server Configuration**.
3. In the right administration pane, click the **Replication** tab.
4. Select **Notification Configuration** from the menu bar.
5. To enable server-initiated push notification:
   a) Select **Enable Push-Based Notification**.

      Unselecting this option disables push synchronization and all related configuration properties.
   b) Enter a time interval in seconds, minutes, or hours to specify the frequency with which the push notification table is polled. The default is 10 seconds.
6. To enable client-initiated push notification:
   a) Select **Enable Pull-Based Notification**.

      Unselecting this option disables pull synchronization and all related configuration properties.
   b) Enter a time interval in seconds, minutes, or hours to specify the frequency with which the pull notification table is polled. The default is 10 seconds.
7. In the Notification Configuration menu, click **Save**.

Setting Up Lightweight Polling for a Single Client

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

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

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

Note: Do not set the client **poll_every** value to a shorter time interval than the server value. This does not result in receiving push notifications any faster, and can cause the client to see the same notification multiple times, causing multiple useless synchronizations. Only set this value on the client if for some reason you do not want to see notifications as frequently as the server checks for pending notifications.

Cache Data Management

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

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

- Data change notifications (DCNs) -- when data used by an application changes on an EIS server, DCNs notify Unwired Server to synchronize cache data. DCNs are useful to cache management, since they facilitate time-sensitive data synchronization between scheduled cache refreshes.
- Cache refresh schedules -- update the cache with the most recent EIS data at regularly scheduled intervals, or on demand. The remote client database eventually retrieves updated data from the server's local copy in the CDB, using one of the supported synchronization triggers. Cache refresh schedules ensure data availability while managing the network traffic required to maintain that data.

Data Change Notifications

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

Developers typically use DCNs because they:

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

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

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

- Users do not always know when to synchronize. Waiting for a user-triggered synchronization is unreliable.

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

You can configure either secure (HTTPS) or non-secure (HTTP) for DCNs. For details on how DCNs are created and sent see the *Server API Cookbook > Server API > Data Change Notification Interface*.

See also

- *Data Change Notification Interface* on page 34
- *Configuring Relay Server Timeouts for DCNs* on page 74
- *Encrypting DCN Connections* on page 101

Cache Refreshes

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

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

Scheduled cache refreshes control the frequency with which objects update data and regulate network traffic by synchronizing data at strategic times, rather than pushing data changes through as they occur. Administrators can also perform on-demand cache refreshes for cache groups at a specified time.

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

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

Aggregate Updates to Multiple MBOs

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

Cache groups specify data refresh behavior for every mobile business object (MBO) within a group. MBOs can belong to only one cache group. Cache groups are created during

development, at which point MBOs are grouped together based on their data refresh requirements. Since all MBOs in a cache group are refreshed simultaneously according to the group's cache policy, cache groups ensure that MBOs with related content remain consistent.

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

See also

- *Schedules to Manage Update Frequency* on page 165
- *Notifications to Update Client Data* on page 166
- *Determining the Cache Interval and Schedule Refresh for a Cache Group* on page 167

Viewing Cache Group Properties

Use Sybase Control Center to review properties for a selected cache group.

Purging a Cache Group

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

1. In the left navigation pane, expand the **Packages** folder and select the package to configure.
2. In the right administration pane, select the **Cache Group** tab.
3. Select the desired cache group and click **Purge**.
4. Enter a **Date** and **Time**. All logically deleted data that is timestamped earlier than this date and time is physically deleted from the cache.
5. Click **OK**.

Schedules to Manage Update Frequency

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

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

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

- Cache Interval – Unwired Server keeps a local copy of enterprise data in the consolidated database (CDB), and uses a synchronization server to manage updates between the CDB

and EIS servers. When data is updated, the remote client database eventually retrieves updated data from this local copy in the CDB. The caching mechanism allows MBOs to retrieve updated data even if back-end servers fail. The cache interval balances the frequency with which the object updates enterprise data with the amount of network traffic required to maintain that data.

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

For example, if the cache interval is 0, each time a client application synchronizes, there is a pause while the Unwired Server rereads data from the EIS and updates the CDB. If, however, the cache interval is greater than 0, then the wait time depends on how long ago the data was refreshed. If the synchronization falls within a recent cache update, synchronization is almost immediate.

- Schedule Repeat – data is refreshed according to a schedule you set up. If you set up a schedule to repeatedly refresh data, information is always refreshed, regardless of the cache interval value. However, typically, you would set a schedule to closely match the cache interval, which is especially good practice for data that changes infrequently or is shared by many clients.

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

See also

- *Aggregate Updates to Multiple MBOs* on page 164
- *Notifications to Update Client Data* on page 166
- *Determining the Cache Interval and Schedule Refresh for a Cache Group* on page 167

Notifications to Update Client Data

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

Messaging-based synchronization (MBS) clients directly receive the unit of changed data for MBOs that belong to the synchronization group. The change detection interval for the

synchronization group determines how often MBS applications push MBO data to the client when cache refreshes occur.

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

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

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

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

See also

- *Aggregate Updates to Multiple MBOs* on page 164
- *Schedules to Manage Update Frequency* on page 165
- *Determining the Cache Interval and Schedule Refresh for a Cache Group* on page 167

Determining the Cache Interval and Schedule Refresh for a Cache Group

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

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

1. Choose the cache group:
   a) In the left navigation pane, expand the **Packages** folder, and select the package.
   b) In the right administration pane, click the **Cache Group** tab.
   c) Select the cache and click **Properties**.
2. Choose an appropriate cache interval. The cache allows you to associate a frequency interval (hour, minute, seconds, and so on) for the cache group's refresh schedule.

For example, if an interval value is set to 0 seconds, then every client synchronization repopulates the entire cache. Instead, choose a value that is longer in duration.

3. Set the repeat for the cycle of intervals.

   Sybase recommends that you choose a cache interval value that supports DCN behavior. See *Sybase Control Center Help > Configure > Configure Unwired Platform > Packages > Configuring a Cache Group > Configuring Scheduled Cache Groups Properties*.

See also

- *Aggregate Updates to Multiple MBOs* on page 164
- *Schedules to Manage Update Frequency* on page 165
- *Notifications to Update Client Data* on page 166

Example Data Update Models

Scenario 1: Product Sales with Expected and Unexpected Changes

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

To support this deployment the administrator might:

1. Evaluate the business context.

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

2. Configure data refresh schedules based on this context.

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

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

3. Anticipate unexpected events and modify the schedule or the cache interval as required.

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

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

See also

- *Scenario 2: Urgent Alerts using Subscriptions and Schedules* on page 169

Scenario 2: Urgent Alerts using Subscriptions and Schedules

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

To support this deployment, an administrator might:

1. Evaluate the business context.
   A hospital executive demands that alerts be delivered to a specific doctor (or medical team) according to each person's unique user name, within one minute of the message being created in the EIS server.
2. Configure data refresh schedules based on this context.
   The Unwired Server administrator responds in such a way that when a new alert is entered into the back-end server, the alert is delivered to the corresponding doctor by setting the cache interval to 0 or "real time". Data is always live and immediate with no caching involved. However, the administrator also sets the schedule repeat for 1 minute and is required 24 hours a day, seven days a week. This refresh schedule is paired with a subscription template for the UrgentAlert MBO with push synchronization enabled.

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

See also

- *Scenario 1: Product Sales with Expected and Unexpected Changes* on page 168

Device and User Management

Device user management applies to broad category of functions: registering devices and managing user accounts. It can also extend to activating and provisioning mobile devices.

Mobile management features of Unwired Platform allow mobile devices and their user accounts to be managed in similar fashion to how computer applications are managed by IT departments: users need to be identifiable, computers need to be tracked and assigned, and "standard" enterprise applications need to be distributed and installed.

In Unwired Platform, the device and user management paradigm includes:

- Device user management

- Device activation and setup
- Remote device provisioning (system infrastructure and applications)
- Subscription services

Device and User Management Overview

The goal of device user management is to register, activate, and provision devices so that device users can access applications and be tracked and managed by the system. This is particularly important for messaging-based synchronization (MBS) environments, because users cannot access the application unless the device is registered and activated.

Table 15. Device and user management tasks

Task	Frequency	Accomplish by using
Activate and deactivate application devices according to the type of synchronization model used (RBS or MBS)	When a new user of a messaging or mobile workflow package must be added	Sybase Control Center for Unwired Platform with the Device Users node.
Provision devices	When a new user of a messaging or mobile workflow package must be added or when a new device is made available to a user	Afaria for advanced or over-the-air capabilities or device cables for development testing. See *Sybase Unwired Platform System Administration > Device and User Management > Device Provisioning*.
Review registered devices and users, delete devices to free licenses, delete users to remove them from the system	As required	Sybase Control Center for Unwired Platform with the Device Users node.
Manage subscriptions	As required	Sybase Control Center for Unwired Platform with the Packages node.
Add and manage device templates	As required	Sybase Control Center for Unwired Platform with the Device Users node.

Users

In Unwired Platform, application users are individuals who have been registered through messaging-based or replication-based applications. Application users are managed in Sybase Control Center.

A user is automatically registered upon first successful authentication by the security provider associated with the package that user is trying to access from a device. A user can have multiple devices.

The platform administrator can view the user name and the security configuration that was used to authenticate the user. The administrator can also review devices associated with a user

to perform any device deletion to free up license. In addition, the administrator can remove users that no longer exist.

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

Messaging Devices

Messaging devices contain applications that send and receive data through messaging. A platform administrator must configure the device activation template properties for messaging-based synchronization (MBS) devices. Device activation requires user registration. Upon successful registration, the device is activated and set up with the template the administrator has selected.

Device registration pairs a user and a device once the user supplies the correct activation code. This information is stored in the messaging database, which contains extensive information about users and their corresponding mobile devices.

Users who are registered but who have not yet installed the software are listed in the window as **registered**, and their messages are queued by Unwired Server for later delivery.

Typically, device registration occurs when the user initially attempts to connect to Unwired Server. However, an administrator can force a user to reregister if there is data corruption on the device, or if the user is assigned a new device. This reestablishes the relationship between the server and the device and refreshes the entire data set on the device.

iPhone devices are listed with other messaging device types. However, unlike other device types, each iPhone application on an iPhone device has its own entry that concatenates the device id with the application shortname.

Note: The device user must activate the messaging account within the number of hours specified in their activation message. If a user does not activate the account within that time frame, an administrator must reregister the user.

See also

- *Always Available* on page 22
- *Message-Based Synchronization* on page 24
- *Subscriptions* on page 174

Device Registration and Activation

Device registration and activation is a required step for a messaging based synchronization environment. These two steps are dependent actions that make the device and the user that is associated with it, part of the Unwired Platform mobility system.

Device registration begins with the registration notice and device template you create once and then reuse in Sybase Control Center. The registration notifies users that they must identify themselves by pairing the user identity and device identity. The device template then sets up

the device and makes it compatible for Unwired Platform messaging use. You can use as many templates as you required.

Once the registration notification is received, the user submits the information back to Unwired Server. This action creates a registration slot that gets occupied by the device upon device activation.

Once the device is activated, the process then opens the communication channel used to relay messages and events to the device. A user can create subscriptions which are then associated with the registered device.

MBS Device Maintenance

As changes occur to the users in your messaging environment, you need to change who is eligible to access data remotely and thereby modify the device state over time.

Use this table to help you assess what administration action to perform from the Sybase Control Center Device Users node when the status of a user or device changes:

Scenario	Device management feature required
A user in the system loses, breaks, or receives a new device.	The **Devices** tab allows you to clone information for MBS devices. Cloning creates a duplicate copy of a device user's messaging configuration settings. This duplication allows you to retain user information but instead pair it with a different device. The administrator can also lock or delete the device, as required.
Synchronization fails and you need to: • Reestablish a relationship between the device and Unwired Server. • Purge existing data from the device. The user fails to register within the allocated time period.	The **Devices** tab allows you to reregister devices. During device reregistration, Unwired Server purges all enterprise data on the device. It retains the device ID and subscription information so that all data can then be resynchronized and loaded onto the device.
A user has multiple devices associated with them, and you need to view those devices.	The User node **Users** tab allows you to open a View Devices dialog. All devices paired with that user ID are listed there.
A device has changed ownership from one user to another.	The User node **Devices** tab allows you to list all devices registered and active with Unwired Server. The administrator can reregister the device and delete the old user.

Scenario	Device management feature required
A user no longer works for your organization or has taken an extended leave. Irrespective of whether the device belongs to the organization or the user, you want this account disabled and thereby not allow the user to access data remotely any longer.	The **Devices** tab allows you disable the entire device account. When you delete a device, all corresponding package subscriptions related to that device are removed from the system, and the license used by that device is returned to the pool of available license slots tracked by Unwired Server. Actual user data, such as personalization keys, is not deleted.
A support request requires an end-to-end investigation of the environment. You need to access the device log in addition to server environment logs.	The **Devices** tab allows you to send a request to Unwired Server to retrieve log files from one or more messaging-based synchronization (MBS) devices.

For details about all of these actions and how they are performed, see *Sybase Control Center online help>Manage>Managing Unwired Platform>Routine Command and Control Actions>Provision>Device Users>Devices*.

Replication Devices

Replication devices are used with replication-based synchronization (RBS) mobile business objects that rely on RBS data cached in the consolidated database. RBS device users are automatically registered when they first synchronize data. There is no device configuration required; the only tasks an administrator performs are monitoring RBS device activity, locking and unlocking RBS devices, and deleting them.

An administrator can lock or unlock devices to disallow or allow users from the device to access the Unwired cluster. All activities, including sending of the push notifications, are stopped while a device is locked. The device application will get an error when trying to communicate with Unwired Server. See *Sybase Control Center > Manage > Managing Unwired Platform > Routine Command and Control Actions > Provision > Device Users > Devices > Locking and Unlocking Devices*.

See also

- *Occasionally Disconnected* on page 22
- *Replication-Based Synchronization* on page 23
- *MBO Package Management Overview* on page 150
- *Subscriptions* on page 174

RBS Device Maintenance

As changes occur to the users in your replication environment, you need to change who eligible to access data remotely and thereby modify the device state over time.

Use this table to help you assess what administration action to perform from the Sybase Control Center Device User node when the status of a user or device changes:

Scenario	Device management feature required
A user in the system loses, breaks, or receives a new device.	None. The registration of the device is automatic for replication environments. This registration occurs during synchronization. The administrator can lock or remove the device, as required.
A user has multiple devices associated with them, and you need to view those devices.	The Device User node **Users** tab allows you to open a View Devices dialog. All devices paired with that user ID are listed there.
A device has changed ownership from one user to another.	The Device User node **Devices** tab allows you to list all devices registered and active with Unwired Server. The administrator can delete the device and the old user from the system.
An RBS device needs to now support MBS applications in addition to RBS applications.	The Device User node **Devices** tab allows send an upgrade request that also allows the user to register the device with Unwired Server as a messaging device. You must then administer the device as both an RBS and MBS device.
A user is taking a short leave and you do not want to change the security configuration required for the package to temporarily disable access.	The **Devices** tab allows you lock and unlock the device account. Lock or unlock devices to control which users are allowed to synchronize data.

For details about all of these actions and how they are performed, see *Sybase Control Center online help>Manage>Managing Unwired Platform>Routine Command and Control Actions>Provision>Device Users>Devices*.

Subscriptions

Subscriptions are managed with MBO packages. However, they are closely associated nonetheless with the device user, because it is the pairing of a user profile and registered and activated device account give the user subscription privileges. Subscriptions define what subset of data the user wants to receive when they synchronize.

Subscription management actions, by whom the action is performed (device user or administrator), as well as the information contained in the subscription, varies depending on the application data required by the MBO package deployed.

Package type	Action	Performed by
Replication	Create subscription template to allow the device user to be notified when information is available, depending on the subsciption configured for that package.	Administrator
	View subscription information, which include the device ID, the device user, and the synchronization group notification threshold, sync counts, and last sync time.	Administrator
	Configure push notifications to activate notification delivery and set the push frequency.	Device user
Messaging (and the DOE-C subtype)	Subscribe to data to determine the frequency of data deliveries as well as other properties associated with messaging synchronization.	Device user
	View subscription information, which includes the device ID, the device user, device status, the last outbound response time, log levels,nd the application name and application client ID that uses the subscription.	Administrator

For information about other management actions you perform, see *System Administration for Unwired Platform> Systems Adminsitration> Package Adminsitration>Managing Deployed Package Subscriptions*. For information about subscription properties, see *Sybase Control Center online help>Configure>Configuring the Unwired Platform> Packages*.

See also

- *Always Available* on page 22
- *Message-Based Synchronization* on page 24
- *Messaging Devices* on page 171
- *Occasionally Disconnected* on page 22
- *Replication-Based Synchronization* on page 23
- *MBO Package Management Overview* on page 150
- *Replication Devices* on page 173

Device Provisioning

Provisioning describes how to setup the device, so it can interact with the Unwired Platform environment. Device provisioning assumes that the user subscriber accounts have already been setup in Sybase Control Center.

Device provisioning is supported differently, depending on various mobility environment factors:

Method	Deployment	Device types	Unwired Platform components
Cradles with desktop device managers	Personal deployment [1]	Windows, Windows Mobile, BlackBerry, Symbian	For runtime clients (messaging/replication) and device applications.
Afaria Over-the-air (OTA)	Enterprise deployment in production environments	Windows, Windows Mobile, BlackBerry, Symbian	For runtime clients (messaging/replication) and device applications. This type of provisioning is typically jointly performed by the administrator (back-end setup to OTA environment) and the device user (installation on the device) after the notification is received.
iTunes	Enterprise deployment with an Apple enterprise certificate	iPhone	For complete packaged iPhone applications that are distributed by businesses to internal users — especially when users do not have an App Store account.

Device provisioning includes:

1. Distributing and installing the platform infrastructure components. Devices that require provisioning of infrastructure components include:
   - Afaria clients for Windows, Windows Mobile, BlackBerry and Symbian devices
   - Client runtime for messaging-based synchronization applications for Windows and Windows Mobile. No runtime is required for iPhone or Symbian.
2. Distributing and installing the device application that interacts with MBO packages that are deployed to the Unwired Server. For supported device types, administrators can use Afaria. For unsupported types (like iPhone), administrators must review the device's user guide to see how application provisioning might occur.

[1] Either development/test or small-scale production environments

3. If you are using a replication base sycnhronization application, setting up the device for push synchronization.
4. If you are using a messaging-based synchronization package, registering the user devices using Sybase Control Center (each user then activates using the Sybase Settings application deployed in step 1).

Runtimes and Clients

In Unwired Platform there is a difference between runtime and clients.

There are two parts to setting up a device as a client to Unwired Platform, depending on the device type and the application you require for your mobile environment:

Client software	Description
Unwired client runtimes and applications	Administrators must provision the Unwired client runtime files and application binaries along with the Afaria client if necessary.
Afaria clients	Afaria clients are included with Afaria. Administrators must download and install one of these clients on the device if you want to provision devices with Afaria over-the-air (OTA); until then, administrators and developers can provision individual devices sequentially by cradling a mobile device and using the devices desktop manager.

The runtime and client files for your device type are located in thhis: `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\ClientAPI\`. There are folders for each runtime and for Afaria client type. Install them files into the device's application folder.

See also

- *Afaria Provisioning and Mobile Device Management* on page 177
- *Apple Provisioning for iPhone* on page 181
- *Setting up Push Synchronization for Replication Synchronization Devices* on page 184

Afaria Provisioning and Mobile Device Management

Afaria extends Unwired Platform functionality by providing additional device client management features for remote and mobile computing devices, including laptops, desktops, and handheld devices.

Use the Afaria tools to:

- Deploy client and runtime infrastructure components over-the-air
- Track assets
- Secure devices and data

See also

- *Runtimes and Clients* on page 177

- *Apple Provisioning for iPhone* on page 181
- *Setting up Push Synchronization for Replication Synchronization Devices* on page 184

Opening the Afaria Administration Console

Use the Afaria® Web interface administration console to manage Afaria resources.

Prerequisites

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

Task

1. From the Sybase Control Center menu, click **Resource** > **Register**.
2. Configure the **Resource Type** and **Connection Information** properties.
   For example, if Sybase Control Center is installed on the same computer as Afaria Server, use the following connection properties:

Options	Description
Host	localhost
Port	80

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

Setting Up the Afaria Environment

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

Prerequisites

Ensure you have configured Sybase Control Center to open Afaria Administrator. No desktop shortcuts are created with an Unwired Platform installation of Afaria.

Be aware of the following Unwired Platform requirements:

- Only use IIS on Windows as the OTA Deployment Center host when using Afaria with Unwired Platform.
- Install the OTA Deployment Center behind the DMZ and use a Sybase relay server to relay download requests.

- Do not install the deployment center on the same host as either Afaria Administrator or Afaria Server; doing so creates a greater risk of a port conflict, depending on protocols used (by default, the Afaria components share port 80).
- Review the system requirements information Afaria provides.

1. *Setting Up the OTA Deployment Center and the SMS Gateway*

   The OTA Deployment Center allows device users to access and download Afaria clients and runtimes over-the-air, rather than needing cradle the device and connect to the network. OTA deployment is recommended for production environments that require you to administer many device users.
2. *Configuring Afaria Server*

   After you have set up OTA Deployment Center and configured either an SMS or SMTP gateway, configure Afaria Server.
3. *Creating Addresses, Groups, and Profiles*

   Session Manager allows you to control each user session opened with the SMS or SMTP download notification and determine what actions are subsequently performed.
4. *Create and Deploy Afaria Clients and Unwired Platform Runtimes*

   When the user receives a client notification, they download the installer and install the Afaria client and any required Unwired Platform runtime files.

Setting Up the OTA Deployment Center and the SMS Gateway

The OTA Deployment Center allows device users to access and download Afaria clients and runtimes over-the-air, rather than needing cradle the device and connect to the network. OTA deployment is recommended for production environments that require you to administer many device users.

1. Install the Afaria SMS gateway, which is a required communication path for the OTA Deployment Center.

   See *Afaria Reference Manual | Platform > Server Configuration > Properties > SMS Gateway Installation*.
2. Set up the OTA Deployment Center.

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

Configuring Afaria Server

After you have set up OTA Deployment Center and configured either an SMS or SMTP gateway, configure Afaria Server.

1. Open Afaria Administrator and configure Afaria Server to use the OTA Deployment Center and SMS gateway:

- **Server Configuration > Properties > OTA Deployment Center** to configure settings for this component. If you are using a relay server, ensure that you configure values used by this Unwired Platform component correctly.

  See *Afaria Reference Manual | Platform > Server Configuration > Properties > OTA Deployment*.
- **Server Configuration > Properties > SMS Gateway** to enable the SMS channel, which is a required communication path for the OTA Deployment Center.

  See *Afaria Reference Manual | Platform > Server Configuration > Properties > SMS Gateway* and *Afaria Reference Manual | Platform > Server Configuration > Properties > Addresses and routing for Afaria messages*.

2. (Optional) Configure an SMTP gateway to send SMS messages over an e-mail infrastructure, which may be used for e-mail-enabled messaging devices.

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

Creating Addresses, Groups, and Profiles

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

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

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

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

   Note: If you have a user list in another application and that application allows you to export entries to a `*.CLV` file, you can import this information into Afaria Administrator. See *Afaria Reference Manual | Platform > Home > Client Deployment > Address Book Properties, Distribution List Properties, and Importing Addresses*.
2. Create groups, and assign users to them.
3. Create profiles that define what happens during the user session (for example, what instructions are given, which items are sent to a client, and so on).
4. Assign these groups (and therefore its members) to the session channel.

Create and Deploy Afaria Clients and Unwired Platform Runtimes

When the user receives a client notification, they download the installer and install the Afaria client and any required Unwired Platform runtime files.

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

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

   CAB files must be signed and placed in the installation sequenced required. Otherwise, the user receives SMS or SMTP messages that link to the OTA Deployment Center in an incorrect order. See *Afaria Reference Manual | Platform > Creating Clients > Creating Afaria Clients*, and refer to the program's context-sensitive help as required.

   Note: If you select the **Companion PC** option, device users can also install by cradling the device and synchronizing the files and seed data from the corresponding desktop device manager (for example, ActiveSync).

2. To make downloads available, publish components to the OTA Deployment Center. Use program called OTA Publisher, which is installed on the Afaria Server host machine in this location: `<AfariaServerInstallDir>\Bin\OTAPublisher.exe`. For details about how to use the OTA Publisher, see the program's context-sensitive help.
3. Send a device notification to allow the user to trigger the provisioning process.

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

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

Apple Provisioning for iPhone

Unlike Afaria, Apple provisioning is primarily performed by developers at the end of their development cycle, with some Sybase Control Center configuration performed by the administrator, and some device setup by the device user.

iPhones do not require a runtime or a client; only the application must be deployed to the device. The lack of a runtime allows users to self-manage their devices: device users download application files to their iPhones and synchronize updates as required. However, to receive notifications, the device users must first enable notifications on the iPhones.

Administrators must configure Apple Push Notification System (APNS) for iPhones. You cannot use APNS on a simulator in a test environment, nor can you use it with iTouch or iPad.

See also

- *Runtimes and Clients* on page 177
- *Afaria Provisioning and Mobile Device Management* on page 177
- *Setting up Push Synchronization for Replication Synchronization Devices* on page 184

Provisioning an iPhone Application with iTunes

To provision the Sybase-packaged iPhone application to your internal users, especially if they do not have an App Store account, use iTunes as an alternative method. You can also use this method if you are building your own iPhone application.

Prerequisites

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

- Sign up for the iPhone Developer Program, which gives you access to the Developer Connection portal. Registering as an enterprise developer gets you the certificate you need to sign applications with.
- Download and customize the default Sybase iPhone application for your Sybase Mobile Workflow or Sybase Mobile CRM solution.
- Create an AppID and ensure that it's configured to use Apple Push Notification Service (APNS).
- Create and download an enterprise APNS certificate that uses Keychain Access in the Mac OS. The information in the certificate request must use a different common name than the development certificate they might already have. This is because the enterprise certificate also creates a private key, which must be distinct from the development key. This certificate must also be imported as a login keychain and not a system key chain and the developer should validate that the certificate is associated with the key in the Keychain Access application. *Get a copy of this certificate.*
- Create an enterprise provisioning profile and include the required device IDs with the enterprise certificate. The provisioning profile authorizes devices to use applications you have signed.
- Create the Xcode project ensuring the bundle identifier corresponds to the bundle identifier in the specified App ID. *Ensure you are informed of the "Product Name" used in this project.*
- Used APNS initialization code in the codeline.

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

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

Task

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

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

If the ports are open, you can connect to the Apple push gateway and receive feedback from it.

2. Copy the enterprise certificate (`*.p12`) to the computer upon which Sybase Control Center has been installed. Ensure it is saved to the `<SUP_Home>\Servers\MessagingServer\bin\` directory.
3. In Sybase Control Center, expand the **Servers** folder and click **Server Configuration** for the primary server in the cluster.
4. In the **Messaging** tab, select **Apple Push Configuration**, and do the following:
   a) Configure **Application name** with the same name used to configure the Product Name in Xcode.
   b) If the certificate is not displaying automatically, browse to the directory.
   c) Change the Push Gateway information to match that used in the production environment.
   d) Restart Unwired Server.
5. Verify that the server environment is set up correctly:
   a) Open `<SUP_Home>\Servers\UnwiredServer\logs\APNSProvider`.
   b) Open the log file that should now appear in this directory. The log file indicates if the connection to the push gateway is successful or not.
6. Deploy the application and the enterprise distribution provisioning profile to your users' computers.
7. Instruct users to use iTunes to install the application and profile, and how to enable notifications. In particular, device users must:
   - Download the Sybase CRM 1.0 application from the App Store.
   - In the iPhone Settings app, slide the **Notifications** control to **On**.
8. Verify that the APNS-enabled iPhone is set up correctly:
   a) Click **Device Users**.
   b) Review the Device ID column. The application name should appear correctly at the end of the hexidecimal string.
   c) Select the Device ID and click **Properties**.
   d) Check that the APNS Device Token has been passed correctly from the application by verify that a value is in the is row. A device token only appears after the application runs.
9. Test the environment by initiating an action that results in a new message being sent to the client.

   If you have verified that both device and server can establish a connection to APNS gateway, the device will receive notifications and messages from the Unwired Server. This includes CRM messages, workflow messages, and any other messages and meant to be delivered to that device. Allow a few minutes for the delivery or notification mechanism to take effect and monitor the pending items in the Device Users data to see that the value increases appropriately for the applications.

10. To troubleshoot APNS, use the `<SUP_Home>\Servers\Unwired Server\log \trace\APNSProvider` log file. You can increase the trace output by editing `<SUP_Home>\Servers\MessagingServer\Data\TraceConfig.xml` and configuring the tracing level for the APNSProvider module to debug for short periods.

Setting up Push Synchronization for Replication Synchronization Devices

If your device application requires push synchronization, you must configure the device for server-initiated synchronization.

Prerequisites

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

- *Configure > Configuring the Unwired Platform Environment > Unwired Server > Server Properties > Replication > Enabling Push Synchronization*
- *Configure > Configuring the Unwired Platform Environment > Packages > Setting Up a Cache group > Scheduling a Cache Refresh*

Task

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

See also

- *Runtimes and Clients* on page 177
- *Afaria Provisioning and Mobile Device Management* on page 177
- *Apple Provisioning for iPhone* on page 181

CHAPTER 8 Systems Maintenance and Monitoring

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

Use monitoring and regular analysis of system logs to:

- 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

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:

- Replication-based synchronization
- Messaging-based synchronization
- System queue status
- Data change notifications
- Device notifications (RBS)
- Package statistics
- Device users
- 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 16. 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

Status and Performance Monitoring

Determine whether your environment is working efficiently and obtain detailed information about the status of the resources in your environment.

The Monitor node in Sybase Control Center is typically used to monitor:

- System diagnostics -- values in the Package, User, Replication, Messaging, and Messaging Queue tabs can help you troubleshoot the system when device clients or applications behave unexpectedly. Information is separated by the package type, either replication-based or messaging-based synchronization. Check to see if synchronization is blocked, during which synchronization phase, and identify the MBO or operation that may be causing a problem.
- Performance indicators -- key performance indicator columns, particularly in the Replication, Messaging, Data Change Notification, and Cache tabs, can help you identify

trends that can be used for tuning your system or determining when to scale up deployed server and data components.

Note: To perform robust data analytics, export information to an XML or a CSV (comma-separated value) file and use an analytic tool of your choosing. See *Sybase Control Center online help > Monitoring > Monitoring Unwired Platform*.

See also

- *Monitoring Unwired Platform* on page 187
- *Reviewing System Monitoring Data* on page 192
- *Monitoring Database Schema* on page 339
- *Setting Up an Existing Database for Monitoring* on page 78
- *Configuring Monitoring Performance Properties* on page 190

Monitoring Unwired Platform

Configure settings to audit the performance and availability of server and application environments.

Monitored operations include 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 TXT or XML file and then use any available reporting or spreadsheet tool to analyze the data.

See also

- *Reviewing System Monitoring Data* on page 192

Monitoring Profiles

Monitoring profiles specify a monitoring schedule for a particular group of packages. The same monitoring schedule can be applied to packages across different domains; similarly, you can select individual packages for a monitoring profile. These profiles let administrators collect granular data on which to base domain maintenance and configuration decisions.

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

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

Planning for System Monitoring

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

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

   If statistics indicate that Unwired Platform is approaching your capacity guidelines, you may want to collect this data more frequently and flush stale data. You can also use the Administration Client API to automate tasks such as exporting and purging of monitoring data. See *System Administration > Administration Client API*.
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. In the left navigation pane, select **Monitoring**.
2. In the right administration pane, select the **General** tab.
3. Click **New** to create a monitoring profile.
4. Enter a name for the new profile.
5. 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.
6. Select **View my selections** to view the packages you selected for the monitoring profile. Unselect this option to return to the package selection table.
7. 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.
8. 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*.
9. 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.

10. 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. 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, 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 (either ASE or SQL Anywhere) 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. For details on setting up the database, see *System Administration > Environment Setup > Database Setup*.

8. Click **OK**.

See also

- *Monitoring Database* on page 14
- *Status and Performance Monitoring* on page 186
- *Monitoring Database Schema* on page 339

Reviewing System Monitoring Data

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

1. In the left navigation pane, select **Monitoring**.
2. In the right administration pane, select one of the following tabs according to the type of monitoring data you want to view:
   - **Replication**
   - **Messaging**
   - **Queue**
   - **Data Change Notifications**
   - **Device Notifications**
   - **Package Statistics**
   - **User Statistics**
   - **Cache Statistics**

See also

- *Monitoring Unwired Platform* on page 187

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.

Replication Statistics

Replication statistics reflect replication-based synchronization (RBS) 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-based synchronization (RBS) provide real-time information about in-progress synchronizations.

Unwired Server monitors RBS requests using these statistical categories:

Category	Description
Package	The package name.

Category	Description
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.
Device ID	The ID number of the mobile device 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** o see aggregated details by by domain, package, and user about past synchronization events for the defined time frame.

Table 17. Detail view information

Synchronization element	Description
Package	The package name.
Device ID	The ID number of the mobile device that participated in the synchronization request.
User	The user associated with the device ID.

Synchronization element	Description
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 18. Summary view information

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

Category	Description
Total Errors	The total number of errors that occurred for the package during synchronization.
Domain	The domain to which the package involved in synchronization belongs.

Replication Performance Statistics

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

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

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

KPI	Description
Total Distinct Package Synchronization	The total number of packages subject to synchronization.
Total Distinct Users	The total number of users who initiated synchronization requests. This value comprises only individual users, and does not count multiple synchronizations requested by the same user.
Average/Minimum/Maximum Sync Time	The average, minimum, or maximum amount of time Unwired Server took to finish a complete synchronization.
Time at Minimum/Maximum Sync Time	The time of day at which the shortest or longest synchronization completed.
Package with Minimum/Maximum Synchronization Time	The name of the package and associated MBO with the shortest or longest synchronization time.
Average/Minimum/Maximum MBO Rows Per Synchronization	The average, minimum, or maximum number of MBO rows of data that are downloaded when synchronization completes.
Average/Minimum/Maximum Operation Replays per Sync	The average, least, or greatest number of MBOs per synchronization received by Unwired Server from a client.
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.

KPI	Description
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-based synchronization (MBS) 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 MBS 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-based synchronization (MBS) provide real-time information about in-progress synchronizations. Because messaging synchronizations progress rapidly, there is typically little pending MBS data available at any given time.

Unwired Server monitors MBS requests using these categories:

Category	Description
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.

Category	Description
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.
Device	The ID number of the mobile device participating in the synchronization.
User	The name of the user associated with the device ID.

Messaging History Statistics

Historical data for messaging-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 by domain, package, and user about past synchronization events for the defined time frame.

Table 19. Detail view information

Data type	Description
Package	The package name.
Device	The ID number of the mobile device 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.

Data type	Description
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 20. Summary view information

Category	Description
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
Total Operation Replays	The total number of operation replays performed by clients during synchronization
Last Time In	The date and time that the last inbound request was received
Last Time Out	The date and time that the last outbound response was sent
Subscription Commands Count	The total number of subscription commands sent during synchronization; for example, subscribe, recover, suspend, and so on

Category	Description
Total Errors	The number of errors that occurred for the package during synchronization
Domain	The domain to which the package involved in synchronization belongs

Messaging Performance Statistics

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

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

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

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

KPI	Description
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.

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). See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Properties > Configuring System Performance Properties*.

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.

Statistic	Description
Time at Minimum/Maximum Queue Depth	The time and date at which the queue reached its minimum or maximum depth.
Type	The direction of message flow: inbound or outbound.
Total Messages	The total number of messages in the queue at one point since the last server reboot.
Bytes Received	The total number of bytes processed by the queue since the last server reboot.
Last Activity Time	The time at which the most recent message was added to the queue since the last server reboot.

Data Change Notification Statistics

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

Monitoring DCN statistics allows you to troubleshoot and diagnose performance issues if, for example, the cache is not being updated quickly enough. These statistics help to identify which packages took longest to process data changes, as well as times of peak performance or strain on the system.

Data Change Notification History Statistics

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

Details about past notification events are organized into these categories:

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

Data Change Notification Performance Statistics

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

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

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

Key performance indicator	Description
Total Notifications	The total number of notifications sent by the enterprise information system to Unwired Server.
Average/Minimum/Maximum Processing Time	The average, minimum, or maximum amount of time Unwired Server took to process a DCN.
Time at Minimum/Maximum Message Processing Time	The time of day at which the shortest or longest DCN processing event completed.
Time of Last Notification Received	The time at which the most recent DCN was received by Unwired Server.
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-based synchronization (RBS) devices. Historical device notification monitoring reports on the packages, synchronization groups, and devices affected by RBS 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
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.
Device ID	The ID number of the mobile device participating in the synchronization request.
Generation time	The date and time that Unwired Server generated the device notification.
User	The name of the user associated with the device ID.

Device Notification Performance Statistics

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

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

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

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

KPI	Description
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.

KPI	Description
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.

KPI	Description
Total Inbound Messages	The total number of messages sent from clients to Unwired Server.
Total Outbound Messages	The total number of messages sent from Unwired Server to clients.
Total Operation Replays	The total number of operation replays performed by clients on mobile business objects (MBOs).
Total Errors	The total number of errors that took place across all synchronizations.
Total Data Push	The total amount of data transmitted from the server to clients.

User Statistics

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

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

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

Replication User Statistics

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

These key indicators monitor replication users:

KPI	Description
Total Synchronization Requests	The total number of sync requests initiated by a client.
Total Rows Received	The total number of rows received during package synchronization.
Total Rows Sent	The total number of rows sent during package synchronization.
Total Bytes Received	The total number of bytes uploaded from clients to Unwired Server.
Total Bytes Sent	The total number of bytes downloaded by clients from Unwired Server.
Average/Minimum/Maximum Synchronization Time	The average, minimum, or maximum amount of time Unwired Server took to complete a synchronization request.

KPI	Description
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.
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.

KPI	Description
Total Data Push	The total number of import data messages.

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.

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
Device ID	The device ID associated with the user
Package	The package the user was attempting to access
Domain	The domain the user was attempting to access

Cache Statistics

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

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

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

MBO Statistics

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

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

Key performance indicator	Description
Cache Group	The name of the group of MBOs associated with this cache activity.
MBO	The name of the single mobile business object associated with this cache activity.
Number of Rows	The number of rows affected by the cache refresh.
Cache Hits	The number of scheduled cache queries that occurred in the supplied date range.
Cache Misses	The number of on-demand cache or cache partition refreshes that occurred in the supplied date range.
Access Count	The number of cache queries that occurred in the supplied date range.
Minimum/Maximum/Average Wait Time	The minimum, maximum, or average duration of cache queries in the supplied date range. This time does not include the time required to refresh the cache in a cache "miss" scenario. Instead Minimum/Maximum/Average Full Refresh Time exposes this data.
Minimum/Maximum/Average Full Refresh Time	The minimum, maximum, or average duration of on-demand and scheduled full refresh activities in the supplied date range.

Cache Group Status Statistics

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

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

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

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

Refining Scope with Filters, Sorting, and Views

You can review any view in the Monitoring node 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.

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.

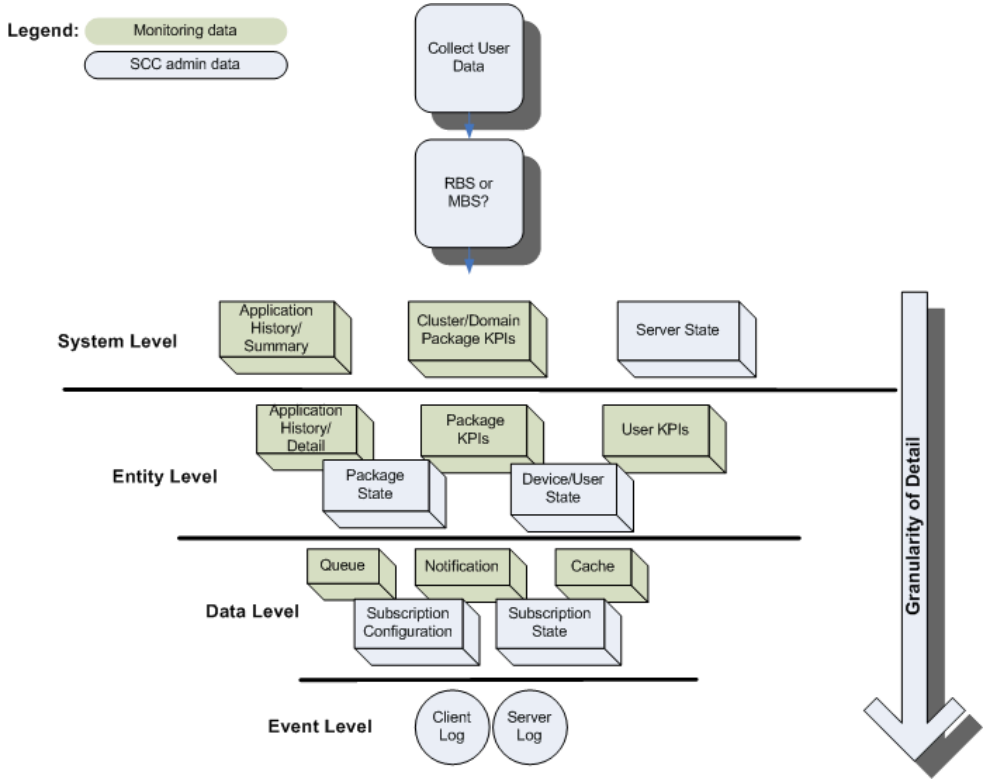
4. Click **Export**.
5. Select a file type for the exported data (CSV or XML), and click **Next**.
6. Click **Finish**.
7. In the file browser dialog, select a save location and enter a unique file name.
8. Click **OK**.
   All monitoring data retrieved by the search is saved to the file you specify in step 7.

System Diagnostics

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 215
- *Access Denied Analysis* on page 219
- *Data Update Failure Analysis* on page 220

Collecting Data

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

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

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

Device Application Performance or Issue Analysis

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

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

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

See also

- *Collecting Data* on page 214
- *Checking Package/User Histories* on page 215
- *Checking Overall Package Statistics* on page 216
- *Checking User KPIs for Other Data Usage* on page 217
- *Checking Server Responsiveness in Sybase Control Center* on page 218
- *Looking for Errors and Failures in SCC* on page 218

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, click the Device Users node, then choose the type of applications that are used on the device:
   - **Unified** for both application types
   - **RBS** for replication applications
   - **MBS** for messaging applications
2. Sort the results by ascending or descending device ID order by clicking the **Device ID** column.
3. Verify whether data is being delivered by looking in the **Pending Items** or **Last Delivery** columns.
4. Expand the **Packages** node, 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, then click the **User Statistics** tab.
2. Click **Security Log**.
3. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the specified time frame.
4. Click the **Result** column to sort rows by result type.
5. Locate any authentication failures or access denied events that are logged for the user who reported the error.
6. If you find any errors in the result column, check package names and security configurations. Then check to see if a similar result is reported for other user names. Also verify if the error persists; a heavily loaded service could cause a transient error. Transient errors are generally resolved retrying the connection.

Next

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

Validating Security Setup

If users are reporting `access is denied` errors where access should be allowed, validate your security setup. A security configuration is defined at the cluster level by the platform

administrator, then assigned at domain and package levels by either administrator type, so it may take some analysis to determine where the problem is occurring.

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

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

Data Update Failure Analysis

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

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

Checking Last Notification Timestamp

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

Prerequisites

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

Task

1. In Sybase Control Center, click the Monitoring node, then click the **Device Notifications** tab.
2. Select **History**.

3. Click **User** to sort on user name in ascending or descending alphabetical order.
4. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the time frame you specify.
5. In the Time of Notification column, ensure that the timestamp was recently recorded, then compare the start and end time of the last known synchronization. Device notification should always occur before a synchronization. If a notification is not received by a user, they may assume that data is not updating correctly.

Checking Cache Statistics

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

1. In Sybase Control Center, click the Monitoring node, then click the **User Statistics** tab.
2. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** so KPIs are aggregated during the specified time frame.
3. Choose the level of cache activity to investigate. Sybase recommends that you:

   - Select **Package level cache group** to investigate cache activity for the cache group, especially if the cache group is used by multiple MBOs. Review the last refresh time to see if data has recently been refreshed from the enterprise information system (EIS), and also check to see that the affected rows are reasonable for the package.
   - Select **Package level MBO** to investigate cache activity at the MBO level. Review data related to cached rows for the MBO, including the number of cache hits and misses. The number of hits should typically be higher than the number of misses. If you do not see any hits, or see a lower number of hits than misses, the notification schedule may not working as designed. See *Validating States and Settings* to see how the subscription that schedules push notifications is set up.

Validating States and 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 state and 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:

- That the group state has a valid status. Cache status can be suspended, available, or refreshing. Set cache group so that it is better coordinated with the change detection interval.
- 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.
   - For replication-based synchronization, that push synchronization is used as required. Follow up with the user to ensure that push synchronization is enabled on the device.
   - That the synchronization group interval is configured appropriately based on the cache interval or schedule repeat. This value determines how frequently change detection occurs.
   - For replicaito-based synchronization, 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 21. Log file locations

Log type	Location
Unwired Server	Aggregated Unwired Server logs, including replication-based synchronization events and SYSAM issues: `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs` Messaging service log details: `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\Data\Log` `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\Trace` Mobile Workflow tracing logs: `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs\WorkflowClient`
Sybase Control Center for Sybase Unwired Platform	Sybase Control Center agent log: `<UnwiredPlatform_InstallDir>\SCC-XX\log\agent.log` Repository log: `<UnwiredPlatform_InstallDir>\SCC-XX\log\repository.log` Request logs: `<UnwiredPlatform_InstallDir>\SCC-XX\services\EmbeddedWebContainer\log\request-<yyyy_mm_dd>.log` Database message: `<UnwiredPlatform_InstallDir>\SCC-XX\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:
	`<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs`
	These log files are generated in the `\logs` directory for the corresponding RSOE:
	• `rsoe.log`
	• `msgrsoe.log`
	• `webserver_rsoe.log`

Installer	`<UnwiredPlatform_InstallDir>`
	`<UnwiredPlatform_InstallDir>\UnwiredPlatform\InstallLogs`
	`<UnwiredPlatform_InstallDir>\UnwiredPlatform\InstallLogs\silentInstall`

Domain	Added to the monitoring database; viewable in Sybase Control Center
Consolidated database	By default, consolidated database errors are logged in the `errorlog.txt` file, located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer`.

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 *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Logs Configuring Server Log Settings*.

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.

Enabling and Configuring Logging

If you are having a problem with either the Unwired Platform runtime or Sybase Control Center administration, you might be able to discover the cause of the problem by enabling or changing the logging level so that more events are recorded.

Configuring Server Log Settings

Configure 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.

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

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

1. In the left navigation pane, expand the **Servers** folder and select the server to configure.
2. Select **Log**.
3. In the right administration pane, click the **Settings** tab.
4. Set the server log size and backup behavior that jointly determine the server log life cycle.
   a) Set the **Maximum file size**, in kilobytes, megabytes, or gigabytes, to specify the maximum size that a file can reach before a new one is created. The default is 10MB.

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

      Alternatively, select **No limit** to retain all log files.
5. For each of the listed components, choose one of these log levels:

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

   The default log levels are:

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

6. 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.

Enabling and Disabling HTTP Request Logging for DCNs

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

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

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

Enabling and Configuring Domain Logging

Activate or deactivate domain logging in Sybase Control Center, and configure domain log autopurge settings for all nodes in a cluster. Domain logging collects data that pertains to the activities of all packages in a domain. You must have administrator privileges to configure domain logging.

First, domain-level logging must be enabled by a platform administrator. Domain-level logging controls whether package-level logging captures data. Then either the platform administrator or the domain administrator can enable logging on a per-package basis from the

Packages node of Sybase Control Center. See *Sybase Control Center online help > Configure > Configuring Unwired Platform > Packages > Enabling Package Logging*.

1. In the left navigation pane, expand the **Domains** folder and select the domain for which to configure log settings.
2. Select **Log**.
3. In the right administration pane, select the **Settings** tab.
4. Select one of:
   - **Enable** – activate domain logging in Sybase Control Center.
   - **Disable** – turn off domain logging.
5. Set the autopurge threshold by entering the length of time (in days) to retain domain log data.
6. Click **Save**.

See also

- *Creating Data Source Connections* on page 144
- *Mapping Roles for a Domain* on page 145

Exporting Domain Log Data

Save a segment of domain log data to a location outside of the monitoring database. Export data to back up information or to perform closer analysis of the data in a spreadsheet application.

1. In the left navigation pane, expand the **Domains** folder and select the domain to configure.
2. Select **Log**.
3. Perform a search to obtain the desired log data.
4. Click **Export**.
5. Select a file type for the exported data (.TXT, 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**.

Configuring Sybase Control Center Logging for Performance Diagnostics

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

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

1. Open `<SCC_HOME>\conf \log4j.properties`.
2. Edit following properties as desired:

- `log4j.appender.agent.File`
- `log4j.appender.agent.MaxFileSize`
- `log4j.appender.agent.MaxBackupIndex`

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

1. Open `<UnwiredPlatform_InstallDir>\SCC-XX\plugins\com.sybase.supadminplugin\agent-plugin.xml`.
2. Add the following line to the file under the `<properties>` element:

   ```
   <set-property property="log_MO_method_execution_time"
   value="enable_log_mo_method_execution_time" />
   ```
3. Open `<UnwiredPlatform_InstallDir>\SCC-XX\conf\log4j.properties`.
4. If you are experiencing log truncation issues, edit the following lines to reflect the required values for maximum file size (default: 5MB) and maximum backup index (20 files):

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

   The `executionTime.log` file now appears in the `<UnwiredPlatform_InstallDir>\SCC-XX\log` folder.

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

1. Modify the `<UnwiredPlatform_InstallDir>\SCC-XX\plugins\com.sybase.supadminplugin\agent-plugin.xml` file as indicated in step 2, above.
2. Restart SCC.
3. Log in and perform your regular administrative tasks.
4. View the execution time indicators for these operations in the cookie file `supatcookie.sol`. The location of this file varies depending on your operating system:

Operating System	Location
Windows XP	`C:\Documents and Settings\<username>\Application Data\Macromedia\Flash Player\#SharedObjects`
Windows Vista	`C:\Users\<username>\AppData\Roaming\Macromedia\Flash Player\#SharedObjects`
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.

Configuring Messaging and Mobile Workflow Runtime Logging

The level of detail in messaging synchronization logs is determined by the log level. You can view this information either in the server log, or you can open the logs files from the `<SUP_Home>\Servers\MessagingServer\Trace` folder.

1. To include messaging runtime log information in the server logs, increase server log levels for the MSG component to capture events beyond the default level of INFO.

   The messaging components that log to the Unwired Server logs are:
   - SUPBridge – used to connect Unwired Server and the messaging synchronization subcomponent.
   - JMSBridge – used for JMS messaging synchronization.
   - MOLogSystem – used to enable system logging across all messaging subcomponents.
2. Alternately, you can manually edit `<SUP_Home>\Servers\MessagingServer\Data\TraceConfig.xml`, to increase the default level of server logging. See *Configuring Mobile Workflow Tracing*.

See also

- *Runtime Message Tracing (TraceConfig.xml) Configuration File* on page 331

Configuring Mobile Workflow Tracing

Set the tracing level for mobile workflow client applications. When mobile workflow tracing is enabled, messages appear as .txt files in *<UnwiredPlatform_InstallDir>*`\`*UnwiredPlatform*`\Servers\UnwiredServer\logs\WorkflowClient`.

1. Open *<UnwiredPlatform_InstallDir>*`\`*UnwiredPlatform-XX*`\Servers\MessagingServer\Data\TraceConfig.xml`.
2. After the last line in the `<TraceConfig>` section, add:

   ```
   <Module Name="Workflow Client" Level="<traceLevel>"
   Desc="<myWorkflowDescription>" />
   ```

   These are the available trace levels:

Level	Messages logged
All	Complete system information
Trace	Finer-grained informational events than debug
Debug	Very fine-grained system information, warnings, and all errors
Info	General system information, warnings, and all errors
Warn	Warnings and all errors
Error	Errors only
Off	Do not record any messages

3. Save the file.
4. Restart the server for the changes to take effect.

See also

- *Runtime Message Tracing (TraceConfig.xml) Configuration File* on page 331

Configuring Messaging Device Logging

By default, messaging device logs are enabled; however, you can configure their behavior from Sybase Control Center. You can retrieve these logs and review them as required.

1. In the left navigation pane, click the **Device Users** node.
2. In the right administration pane, click the **Registration Templates** tab.
3. Set up the advanced properties. For logging, you can set:
   - Trace size
   - Trace level
   - Number of log items recorded

See *Sybase Control Center > Configure > Configuring Unwired Platform > Device Users > Device Templates > Messaging Device Advanced Properties*.

Retrieving Device Logs

Send a request to Unwired Server to retrieve log files from a messaging-based synchronization (MBS) device.

Log file retrieval is supported only for messaging devices.

1. In the left navigation pane of Sybase Control Center, select **Device Users**.
2. In the right administration pane, click the **Devices** tab, and select **MBS** to view messaging-based synchronization devices.
3. Select a device, and click **Get Trace**.

   The status of the device must be either "online" or "offline" to retrieve the log.
4. Click **OK**.
5. When the device is online, open the `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\Data\Client Trace` folder to view the device log.

   Trace files, which can be either .log or .txt file types, have the following file name structure: *yyyyMMddHHmmss_UserName_DeviceName_FileName*. For example: `20091217103050_User1_Emulator67215793_Messaging_xce.log`.

Configuring Relay Server Outbound Enabler 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.

1. Open `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\config\relayserver.properties`.
2. Locate the `RSOE Log Setting` section.
3. Change the value of `relayserver.rsoe.log.level` from 0 to either:

   - 1 – session-level logging. Provides a high-level view of a synchronization session.
   - 2 – request-level logging. Provides a more detailed view of HTTP requests within a synchronization session.
4. Save the file.
5. Ensure the consolidated database is started.
6. Regenerate the service:

   run <UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\bin\regRelayServer.bat install [manual | auto]

See also

- *Relay Server Setup* on page 62
- *Relay Server Documentation* on page 63
- *Relay Server Configuration Files* on page 336
- *Relay Server Utilities* on page 297

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. See *Relay Server (rs.config) Configuration File*.
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.

See also

- *Relay Server (rs.config) Configuration File* on page 336

Enabling Custom Log4j Logging

Use any editor (text, XML, or an IDE) to create a `log4j.xml` file.

Prerequisites

Understand the use restrictions for log4j logging.

Note: The file format of this file falls outside the scope of this document. For details about `log4j.xml` format, see *http://logging.apache.org/log4j/*.

Task

1. In an editor, create a `log4j.xml` file.
2. Define the appenders you require and save the file. Appenders must be named, have a class defined, and require one or more parameters.
3. Add the file to all servers in your cluster. The file must be saved in: `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\`.
4. Repeat these steps on each server in the cluster.
5. Restart all servers.

Log4j Restrictions

The default Unwired Server runtime logging and log4j logging are peer systems; the implementation of one does not usually impact the other, unless functionality overlaps.

Border conditions present in the `log4j.xml` configuration file may interfere with the configuration defined by `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\logging-configuration.xml`. Do not create appenders that output to:

- Any console (either System.out or System.err) with any appender, including ConsoleAppender. Log data destined to these appenders is ignored by Unwired Platform runtime components.
- The Unwired Server log. By default you can find the log file is created as `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\logs\<hostname>-server.log`.

Any changes you make to a `log4j.xml` configuration are applied only to the node on which the change is made; they are not propagated across the entire cluster. Therefore, you must edit the file on each node individually, as required.

Backup and Recovery

Backup and recovery strategies are part of a larger availability and resiliency strategy you created when designing your network for Unwired Platform environment.

- Availability describes the degree of healthy system operations you can maintain under adverse conditions (for example, if multiple nodes in an Unwired Server cluster become unavailable due to a hardware failure).
- Resiliency describes how quickly healthy system operations return after service degradation.

Backup and recovery strategies can be one of two types: error correction and disaster recovery. The Unwired Platform documentation discusses error correction. For disaster recovery, you

may need to engage with other vendors that provide solutions geared to maintaining the viability of your entire enterprise.

See also

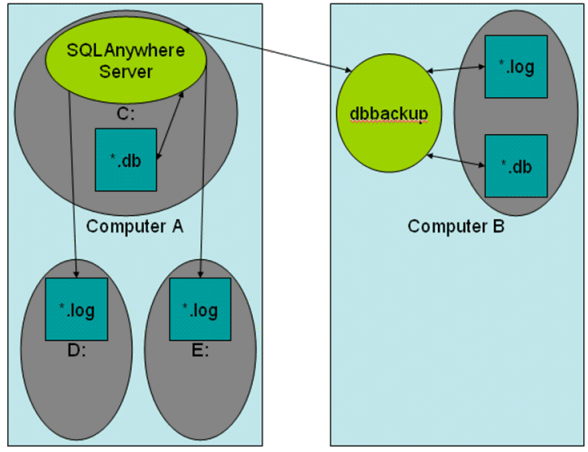
- *Data Management Overview* on page 158
- *Data Mobility* on page 19
- *Database Clusters* on page 48

Sample Backup and Recovery Plan

Provides a basic backup and recovery plan.

This diagram shows the architecture for a reasonably reliable backup and recovery strategy. Only Sybase Unwired Platform components related to database recovery are included.

Figure 2: Sample backup and recovery plan



Shown in the diagram:

- Computer A is where the SQL Anywhere database server is installed and runs under Sybase Unwired Platform. There are three physical disks on this computer:
  - The C: drive has the SQL Anywhere server and the database files (`default.db`, `clusterdb.db`), which hold critical data that Sybase Unwired Platform requires to function.
  - The D: and E: drives hold identical copies of the transaction log files (`default.log`, and `clusterdb.log`). Using SQL Anywhere terminology, the D: drive holds the regular transaction log, and the E: drive holds the mirrored transaction log.

- Computer B is for long term backup, and requires only one drive (or backup tapes). Run the **dbbackup** utility from this computer periodically to obtain a full backup of the `*.db` and `*.log` files from Computer A.

Failure and Recovery Scenarios

Describes several failure scenarios (using the Sample Backup and Recovery Plan setup), how recovery works, and implications for Sybase Unwired Platform operation.

*Disk C has an unrecoverable error. The *.db files have been lost.*
Recovery: Install a replacement disk, and use standard file restore procedures to reinstall the SQL Anywhere software, and whatever else is needed. If the restore returns the `*.db` files, there is no harm, but do not rely on these files to be valid. Instead, copy the last backup version of the `*.db` files across from Computer B.

Next, start the SQL Anywhere server, which detects that the `*.db` files are not up-to-date with the checkpoints in the `*.log` files on drives D: and E: (which are unaffected by the C: drive failure). The server automatically replays transactions recorded in the transaction log to bring the database back to the state of all committed transactions at the time of the C: drive failure.

Sybase Unwired Platform can then start up and run normally. Sybase Unwired Platform mobile device clients are not affected except by the inability to sync between the time of the failure, and the time at which the recovery process has completed.

*Disk D: or E: failure. One of the *.log files has been lost.*
Recovery: Install a replacement disk and restore from backups.

Once the disk has been restored, copy the `*.log` files from the drive that did not fail to the one that failed. Restart the failed drive.

Complete failure of Computer A, and disks lost.
Recovery: See Restore the Consolidate Database for instructions. This should be a very infrequent event.

In this scenario, the database has lost all transactions since the previous backup to Computer B. Any Sybase Unwired Platform mobile device clients that synchronized between the time of the previous backup and the time of the failure cannot now sync. Clients must delete their UltraLite database and start fresh. Any pending operations on these clients are lost. Clients that have not synchronized since the previous backup are unaffected.

Backing Up the File System

Regularly perform backups of the Sybase Unwired Platform files and directories.

Avoid backing up individual artifacts. Instead, Sybase recommends that you perform regular backups of entire directories as part of your disk backup schedule.

Note: At the same time the folder and disk backup is performed, update the Windows registry.

1. Plan the frequency of the file system backups to coincide with any changes made to the system, including:
   - Metadata changes (such as deployment of new mobile business object packages to the server)
   - Configuration changes (such as new enterprise information system connection)
2. Back up application artifacts, by using Sybase Control Center to export packages. This preserves each deployed package in an alternate location. See *Sybase Control Center online help > Manage > Managing Unwired Platform > Routine Command and Control Actions > Deploy > Exporting Package Content*.
3. Back up the contents in these Unwired Server directories:
   - `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin`
   - `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\config`
   - `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\logs`
   - `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository`
4. Back up the contents of your system servers:
   - `<UnwiredPlatform_InstallDir>\Servers\Advantage910`
   - `<UnwiredPlatform_InstallDir>\Servers\MessagingServer`
   - `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11`
5. Back up the contents of these Sybase Control Center directories:
   - `<UnwiredPlatform_InstallDir>\SCC-X_X\conf`
   - `<UnwiredPlatform_InstallDir>\SCC-X_X\services\repository\repository.db`
   - `<UnwiredPlatform_InstallDir>\SCC-X_X\log`

Next

To maintain a consistent backup state, Sybase recommends you back up the data consolidated database at these times as well. See the next task (Backing Up System Data) in this sequence.

Backing Up System Data

In mobility environments, data backups are a fundamental method of securing data. Depending on your database type, different backup mechanisms exist.

For platform data, back up synchronization repositories and Sybase Control Center (SCC) repositories. SCC and the CDB (for replication-based synchronization) can use the process described for SQL Anywhere databases. Messaging data requires its own process.

Note: When you make a backup, decide where to store the backup files: on the Unwired Server host or on some other computer or third-party hardware/software package used for backup purposes.

1. *Backing Up a SQL Anywhere Database*

   Back up platform data warehoused in SQL databases using SQL Anywhere utilities.
2. *Backing Up Messaging Data*

   Use the adsbackup utility to back up messaging data. You can then use MORecover to recover the data to an existing database.

Backing Up a SQL Anywhere Database

Back up platform data warehoused in SQL databases using SQL Anywhere utilities.

Prerequisites

On backup hosts, verify that SQL Anywhere software is installed, and the PATH is set. If you are running Sybase Unwired Platform as a cluster, SQL Anywhere is installed under the Sybase Unwired Platform installation directory. Otherwise, install another copy of Sybase Unwired Platform on backup machines. Otherwise, SQL Anywhere components may not work as expected.

By default, Unwired Platform uses SQL Anywhere for the consolidated and cluster databases. `clusterdb` is present if Unwired Server has been configured to run as a cluster.

Back up these components, which jointly make up a SQL Anywhere database:

- Transaction logs – these files use the extension of `*.log`.
- The database file – this file uses the extension of `*.db`.

See also

- *Backing Up Messaging Data* on page 240

Validating and Backing Up Platform Data

Regularly perform a remote backup of data so it can be recovered. Before you back up system data, ensure the data is valid.

SQL Anywhere has various utilities to help you with these tasks:

- **dbvalid** – validates the integrity of the database by checking the index keys on some or all of the tables in a database, as well as on the database file itself.
- **dblocate** – locates any SQL Anywhere database servers (consolidated or cluster) running over TCP/IP on the immediate network, and prints a list of the database servers and their addresses. This list includes alternate server names. Depending on your network, it may take several seconds for **dblocate** to print its results.
- **dbbackup** – makes a backup copy of all files for a single database from a remote backup location. A simple database consists of two files: the main database file and the transaction log. If you mirrored the transaction log you need not back up this file. All backup file

names are identical to database file names. The image backup contains a separate file for each backed-up file.

Note: Depending on the design of your production environment, the **dbvalid** and **dbbackup** commands may not work remotely.

The database server is named <*clustername*>_primary.

---

1. Validate the databases:
   a) Ensure there are no active connections to the server.
   b) Validate the consolidated database:

      ```
      dbvalid.exe -c "DBF=default.db;UID=dba;PWD=SQL"
      ```

   c) Validate the cluster database:

      ```
      dbvalid.exe -c "DBF=clusterdb.db;UID=dba;PWD=SQL"
      ```

2. On the backup machine, verify that SQL Anywhere software is installed, and the PATH is set.
3. Back up the databases to archive the system data:
   - For consolidated databases, run:

     ```
     dbbackup -c
     "ENG=<clusterName>_primary;DBN=default;UID=dba;PWD=SQL"
     \SQLAnybackup
     ```

   - For cluster databases, run:

     ```
     dbbackup -c
     "ENG=<clusterName>_primary;DBN=clusterdb;UID=dba;PWD=SQL"
     \SQLAnybackup
     ```

   This creates `defaullt.db`, `default.log`, `clusterdb.db`, and `clusterdb.log` in the `\SQLAnybackup` directory on the backup computer.
4. As a precaution, validate the backups are suitable for recovery:
   a) On the backup computer, create a temporary working directory (such as `\tmp`).
   b) Under the temporary directory, create an identical directory structure for the two log locations. You may need to use the **subst** command to map local directories to drive letters used on the runtime computers to the backup location.
   c) Copy `*.log` to these locations.
   d) Run **dbvalid** on the `\tmp` copy of the `.db` file.

      WARNING: Do not run **dbvalid** on the backup copy itself (in the `\SQLAnybackup` directory of this example). The command runs, but corrupts your `.db` file so it cannot be used in recovery.
      - If validation succeeds, the backup in `\SQLAnybackup` can be used for recovery; delete the files in the `\tmp` and log directories.
      - If validation fails, the backup is not usable for recovery; try again.

Next

With the archive of the database complete, you can optionally back up the archive to a tape drive.

Backing Up Messaging Data

Use the **adsbackup** utility to back up messaging data. You can then use MORecover to recover the data to an existing database.

The backup target can be any folder. A collection of files constituting the backed-up data is created in a folder you specify.

Note: Backup files only include aw data in the original tables, without the index data. Therefore, they cannot be used directly as a database and must be recovered with MORecover before they can be used.

You can schedule this command to run regularly when your system is lightly loaded, since the backup operation runs simultaneously with the messaging server's normal processing.

1. Change to `<UnwiredPlatform_InstallDir>\Servers\AdvantageXXX\Server`.
2. Run the **adsbackup** utility to create a MORecover snapshot; for example:

   ```
   adsbackup.exe
   iAPPLICATIONS,CFG_IDS,CFG_PROP_VALUES,CFG_SUBFOLDER_PROP_V
   ALUES,CFG_TEMPLATES,DEVICES,USER_DEVICE,USERS "C:\Sybase
   \UnwiredPlatform-X_X\Servers\MessagingServer\Data\OBR
   \OBR.add" <MYBackupTarget>
   ```

See also

- *Advantage Database Server Backup (adsbackup) Utility* on page 314
- *Backing Up a SQL Anywhere Database* on page 238

Restoration of the Installation File System

Restore the Sybase Unwired Platform installation file system from a backup.

To perform a normal restoration, use the file or disk backup utilities used to perform the backup. Sybase recommends that you save the current installation directory before you restore from backup.

Note: You may also need to restore the Windows registry from the backup done at the same time.

Restoration of the Consolidated Database

Restore the Sybase Unwired Platform consolidated database and transaction logs from a backup.

As discussed in *Failure and Recovery Scenarios*, if only one of C:, D:, or E: drives fail, recovery should be automatic once you have completed the appropriate tasks.

These steps are required in case of complete failure of all three drives:

1. Restore the computer's C:, D:, and E: drives from backup.
2. Delete the `*.db`, `*.log` files from their normal places after you have restored the file system.
3. Copy the `*.db` and `*.log` files from Computer B's backup directory to the normal locations on Computer A.

   Note: Copy the *.log files twice—once to the normal transaction logs directory, and once to the mirrored transaction logs directory.
4. Restart Sybase Unwired Platform.
5. If there have been package deployments or other cluster-affecting operations since the last database backups, the file-system data corresponding to the packages may be out-of-sync with the database contents related to these packages. If this has occurred, the Sybase Unwired Platform servers cannot fully start. Check server bootstrap log and the <hostname>-server.log file. These logs include server mismatch messages that prevents server from starting normally. To recover from this:
   a. Choose one of the Unwired Servers.
   b. Shut down that server.
   c. Edit the `sup.properties`, and change the `cluster.version` property value to match that of the current cluster as reported in the logs. This file is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer`.
   d. Run `updateProps -r` from the same directory to apply the change into the `clusterdb`.
   e. Restart that Unwired Server.

   Note: This server should be able to take over as the Sybase Unwired Platform primary. Sybase recommends you redeploy all your packages (using the **UPDATE** option) to make sure all the packages you expect to be available really are. All of your Sybase Unwired Platform clients should delete their UltraLite databases, and perform full synchronizations.

Restoration of the Messaging Data

Restore the Sybase Unwired Platform messaging data tables from a backup by running the MORecover utility.

You can find this utility in the `<UnwiredPlatform_InstallDir>\Servers\Messaging\Server\Bin` folder.

Before running MORecover to restore your database, convert the data files created by **adsbackup** back to a standard database format, by running **adsbackup** with the -r option. You can then use MORecover to restore the backed-up data.

For example, if you sent your backed-up data output to `c:\bak`, but restored the data to `c:\bak\restore`, the command line syntax for **adsbackup** is:

```
adsbackup.exe -r "c:\bak\obr.add" "c:\bak\restore\obr.add"
```

When you run the MORecover utility, use the restore location. For example:

```
MORecover "c:\bak\restore\obr.add"
```

See also

- *Unwired Server Database File Recovery (MORecover) Utility* on page 316

Platform Licenses

Platform license types are: Personal Developer Edition, Enterprise Developer Edition, or Enterprise Deployment Edition.

The license files for Unwired Platform are installed in the `<sup_home>\Servers\UnwiredServer\licenses` directory. Unwired Server reads the license files in this location.

You can choose from two licensing models; the model you select affects the manner in which your licenses are maintained:

- Unserved license – licensing details are stored directly in the license file. Typically, administrators choose this option when they install all components on a single host, for example, in a Personal Developer Edition or trial scenario. Sybase does not recommend this model for production environments—especially clustered environments, due to the way in which device licenses are tracked and checked out.
- Served license – requires an additional server to manage licensing details for Unwired Platform components. Typically, administrators choose this option in a distributed production environment. Sybase recommends that you use the served license model for shared development environments and production environments; this model is required in a cluster environment.

For details about obtaining a license and installing it with the Sybase Unwired Platform installer, see *Sybase Unwired Platform Installation Guide > Planning Your Sybase Unwired*

Platform Installation > Obtaining a License. For information about SySAM licensing, see the *Sybase Software Asset Management Users Guide* on the Sybase Product Manuals Web site at *http://sybooks.sybase.com*.

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 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 value cannot be retrieved 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 is used to enumerate each device connection made across clustered servers. Every server then checks out all device licenses when the servers start.

Afaria Licenses

Afaria is not licensed by SySAM. Instead, Afaria uses an internal mechanism to track and enforce licenses and features.

The Afaria server included with Unwired Platform uses the Afaria license string for session manager support. The license string is a fixed serial number that determines what options are included. If you require more options than what are currently included with Unwired Platform, upgrade Afaria by ordering additional Afaria options.

For example, if you are using over-the-air (OTA) deployments to support Unwired Platform, you may want to purchase the Outbound option. Otherwise, the Maximum Simultaneous Notifications data element defaults to 20, and you cannot increase the notifications you can use in your environment.

License details appear on the Afaria Administrator License page. The Licensing page contains information about your Afaria system, including a list of licensed components and Afaria client types, the number of licensed sessions, expiration dates (if any), and a brief description of the license type. The information on this page is read-only; you cannot modify any values.

You can also track software compliance as needed. The License compliance view allows you to examine software license compliance and usage data collected via Inventory Manager scans. See *Afaria Reference | Platform > Data views > Tracking Software Compliance and Usage Data*.

License Validation

Attributes in the license file control the base number of devices that can be registered, the number of servers (typically for clustered production environments) you install, and expiry dates for both devices and servers. The mechanism that counts device licenses varies, depending on your model.

There are two licensing models you can use with Unwired Platform:

- Unserved (local) license – uses a local license file for each Unwired Platform installation.
- Served (SySAM License Server) – uses a SySAM License Server to support multiple Unwired Platform installations.

For both models, Unwired Server always tracks available licenses and expiry dates, and writes license errors to the Unwired Server log. Administrators can always check these limits and take appropriate action when that limit is reached.

Unserved model

In an unserved license model, licenses are validated at several intervals:

- At Unwired Server start-up – When Unwired Server starts up, it always checks the license file for the number of servers licensed. If the value cannot be retrieved, or if the server is not licensed, Unwired Server stops.
- At device connection – when the device user tries to connect to Unwired Server, Unwired Server checks the device ID against the data tier. If the device falls within the device license limit, the device connection continues and operations proceed normally for both replication and messaging applications. If the device falls outside the limit, Unwired Server throws a license check exception to the client. For details about deleting unused device IDs to free licenses, see *System Administration Guide > Systems Maintenance and Monitoring > Platform Licenses > Device User License Limits*.
- Upon license expiry – if the date in the license file matches the current date, the license expires; Unwired Server generates a license expired error. The error text varies, depending on whether the server or the client connection licenses have expired. If a server license is expired, Unwired Servers also stop.

Served Model

In a served license model, licenses are validated at these intervals:

- At Unwired Server startup – When Unwired Server starts up, it always checks the license file for the number of servers licensed. If the value cannot be retrieved, or if the server is not licensed, Unwired Server stops.
- With each synchronization – the procedure varies slightly depending on the synchronization model used on the client:
  - For replication-based synchronization – after the device user first attempts to connect to Unwired Server and is authenticated, Unwired Server uses the device ID to check the license into the data tier. If the device falls within the device license limit,

synchronization proceeds. If the device falls outside the limit, Unwired Server throws a license check exception to the client.

Administrators must monitor licenses carefully; there may be many devices connected to the server, but fewer licenses being used. For details about deleting unused device IDs to free licenses, see *System Administration Guide > Systems Maintenance and Monitoring > Platform Licenses > Device User License Limits*.

- For messaging-based synchronization – when the device user tries to connect, Unwired Server checks the device ID against the the data tier. If the device is registered, and the total number of devices registered falls within the device license limit, the message is processed normally. If the device is not registered, or the total number of devices registered falls outside the limit, Unwired Server throws a license check exception to the client.

- Upon license expiry – if the license expires, Unwired Server generates a license expired error. The error varies, depending on whether the server or the client connection licenses have expired. When a server license expires, Unwired Servers also stop.

Device User License Limits

Licenses limit not just how many components you can install on a network, but how many users can connect to your servers.

If you notice messages similar to these errors in the Unwired Server log, then connection requests from registered users have exceeded the licensed limit:

```
2009-12-28 18:01:59.872 INFO    MMS          Thread-19
[com.sybase.sup.server.lm.LicenseUtil] The number of registered
devices has reached the maximum limit for your license.
2009-12-28 18:01:59.965 INFO    MMS          Thread-19
[com.sybase.djc.mobilink.LoginHandler] The number of registered
users has reached the maximum limit for your license.
2009-12-28 18:02:00.168 ERROR   Mobilink     Thread-19
[com.sybase.ml.sup.Logger] [-10052] authenticate_parameters scripts
return 4000
```

For example, a trial license limits you to only five device users. If a sixth user tries to connect, the error is logged accordingly.

In cases where the number of users in your environment exceeds that of your license, you can either:

- Upgrade your license and manually change the license in your environment.
- Control the number of device user connections at a given moment in SCC. For example, you can view the total number of users in the User Statistics tab of the Monitor node.

  If the number of device users is too high for your license, you can manually delete unused devices from the system in the Devices node. See *Sybase Control Center online help > Manage > Managing Unwired Platform > Routine Command and Control Actions > Provision > Device Users > Devices > Deleting Replication and Messaging Devices*.

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 tranferred to new devices.

Unwired Platform licensing is configured during installation. However, if necessary, license details can be changed at a later time. See Sybase Unwired Platform Installation Guide > Postinstallation Tasks > Upgrading License Files.

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:
   - Production edition – the edition of the software you have installed.
   - 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. See *Sybase Unwired Platform Installation Guide > Planning Your Sybase Unwired Platform Installation > Sybase Unwired Platform Licenses > Device User License Limits*.
   - Used device license count – the total number of licenses currently attached to devices. 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. See *Sybase Unwired Platform Installation Guide > Planning Your Sybase Unwired Platform Installation > Sybase Unwired Platform Licenses > Device User License Limits*.
   - 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.
   - 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.
   - Server license type – the type of license currently used by Unwired Platform. For more information on license types, see *Sybase Unwired Platform Installation Guide > Planning Your Sybase Unwired Platform Installation > Sybase Unwired Platform Licenses*.
   - 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.
4. Click **Close**.

Manually Updating and Upgrading Licenses

You must upgrade Unwired Platform and Afaria licenses separately. For Unwired Platform upgrades, the procedure varies depending on whether you are using a served or unserved model.

To update or upgrade your license, ensure you have the updated or upgraded license file and save it in the appropriate location for your served or unserved license model. Then you need to run `license.bat` and restart Unwired Servers for license change to take affect.

1. *Updating and Upgrading Unwired Platform Licenses*

   The <UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin\license.bat script allows you to upgrade Unwired Platform licenses without having to re-run the installer. You can use this script in both served and unserved license models.

2. *Upgrading Afaria Licenses*

   Various mechanisms in Afaria Adminstrator allow you to change the serial number to support additional options and features.

Updating and Upgrading Unwired Platform Licenses

The `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin\license.bat` script allows you to upgrade Unwired Platform licenses without having to re-run the installer. You can use this script in both served and unserved license models.

Prerequisites

For unserved models, ensure that you have already copied the new license file to the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\licenses` folder.

Task

1. Use the license to locate information that the `license.bat` script requires. See *Locating Information in a License File*.

   When you set up your license model, Sybase license fulfillment generates a new Unwired Platform license that includes a new serial number. For subsequent upgrades, use the same serial number to incrementally update licenses.

2. From the command prompt, change to the `bin` folder for Unwired Server.
   For example, `cd C:Sybase\UnwiredPlatform\Servers\UnwiredServer\bin\`.

3. Run `license.bat`:

   `license.bat PE LT <licenseNumber>`,

where PE is the Production Edition, LT is the License Type, and <licenseNumber> are the number of licensed devices supported.

See also

- *Locating Information in a License File* on page 248
- *License Upgrade (license) Utility* on page 307
- *Upgrading Afaria Licenses* on page 250

Locating Information in a License File

After you download a license file, you must extract some information from it to complete your installation.

1. Use a text editor to open your license file.
2. Locate the uncommented line that begins with INCREMENT SUP_BASESRVR.

   For example:

   ```
   ...
   INCREMENT SUP_BASESRVR SYBASE 2010.03316 31-mar-2010 uncounted \
       VENDOR_STRING=PE=EE;LT=CP HOSTID=ANY ISSUER="CO=Sybase, \
   ...
   ```

3. Determine whether the server license is served or unserved.

   If the line beginning with INCREMENT SUP_BASESRVR ends with "uncounted" it is an unserved license. For example:

   ```
   ...
   INCREMENT SUP_BASESRVR SYBASE 2010.03316 31-mar-2010 uncounted \
       VENDOR_STRING=PE=EE;LT=CP HOSTID=ANY ISSUER="CO=Sybase, \
   ...
   ```

   If that line ends with a number immediately following a date, it is a served license. For example:

   ```
   ...
   INCREMENT SUP_BASESRVR SYBASE 2010.03316 31-mar-2010 10 \
       VENDOR_STRING=PE=EE;LT=CP HOSTID=ANY ISSUER="CO=Sybase, \
   ...
   ```

4. Determine the product edition and license type for the license.

   For both served and unserved licenses, note the value of PE (product edition) and LT (license type) in the line following the SUP_BASESRVR line. For example:

   ```
   ...
   INCREMENT SUP_BASESRVR SYBASE 2010.03316 31-mar-2010 uncounted \
       VENDOR_STRING=PE=EE;LT=CP HOSTID=ANY ISSUER="CO=Sybase, \
   ...
   ```

   The PE value is the license product edition value; "EE" in the example above.

   The LT value is the license lype value; "CP" in the example above.

5. Determine the number of client licenses.

a) Locate the uncommented line, beginning with INCREMENT SUP_BASECLIENT.

   For example:

```
INCREMENT SUP_BASECLIENT SYBASE 2010.03316 31-mar-2010
uncounted \
       VENDOR_STRING=PE=EE;LT=ST HOSTID=ANY ISSUER="CO=Sybase,
\
         Inc.;V=1.5;AS=A;MP=365;CP=100;" ISSUED=04-sep-2009
NOTICE="For \
...
```

b) Determine whether the client licenses are served or unserved.

   If the line beginning with INCREMENT SUP_BASECLIENT ends with "uncounted" the client licenses are unserved. For example:

```
INCREMENT SUP_BASECLIENT SYBASE 2010.03316 31-mar-2010
uncounted \
       VENDOR_STRING=PE=EE;LT=ST HOSTID=ANY ISSUER="CO=Sybase,
\
...
```

   Note the value of CP (Client Places) two lines below the SUP_BASECLIENT line.

   If that line ends with a number immediately after a date, the client licenses are served. For example:

```
INCREMENT SUP_BASECLIENT SYBASE 2010.03316 31-mar-2010 100 \
       VENDOR_STRING=PE=EE;LT=ST HOSTID=ANY ISSUER="CO=Sybase,
\
...
```

c) Determine the number of client licenses.

   For unserved client licenses, the number of client licenses is the value of CP two lines below the line beginning with INCREMENT SUP_BASECLIENT. For example:

```
INCREMENT SUP_BASECLIENT SYBASE 2010.03316 31-mar-2010
uncounted \
       VENDOR_STRING=PE=EE;LT=ST HOSTID=ANY ISSUER="CO=Sybase,
\
         Inc.;V=1.5;AS=A;MP=365;CP=100;" ISSUED=04-sep-2009
NOTICE="For \
...
```

   For served client licenses, the number of client licenses is the value at the end of the line beginning with INCREMENT SUP_BASECLIENT. For example:

```
INCREMENT SUP_BASECLIENT SYBASE 2010.03316 31-mar-2010 100 \
       VENDOR_STRING=PE=EE;LT=ST HOSTID=ANY ISSUER="CO=Sybase,
\
...
```

When you run the Unwired Platform installer, enter the information you have extracted from your license file on the license information page.

See also

- *Updating and Upgrading Unwired Platform Licenses* on page 247
- *License Upgrade (license) Utility* on page 307

Upgrading Afaria Licenses

Various mechanisms in Afaria Adminstrator allow you to change the serial number to support additional options and features.

The server configuration area of the Afaria Administrator lets you define parameters for the Afaria Server, including new license data.

1. Obtain a new serial number.

   Sybase license fullfilment generates an Afaria license string that includes a new serial number. For subsequent option upgrades, use the same serial number to incrementally update licenses.
2. Define new software licenses in Afaria Administrator.

   The License Compliance page appears empty until you define licensing details. Once you have defined these software licenses, the page shows data for Client category, Manufacturer, Application, Size, Version, # (number) Purchased, Effective and Expiration dates, and any Notes you add. Initially, licenses are sorted by Client category, Manufacturer, then Application. See *Afaria Reference | Platform > Server Configuration > License Compliance*.

See also

- *Updating and Upgrading Unwired Platform Licenses* on page 247

CHAPTER 9 Administration Client API

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

The client you create connects to Unwired Server via Sybase Control Center embedded management server and Sybase Unified Agent. For details, see *Administration Client Developer's Reference*.

The administration client API includes these interfaces:

Interface	Includes methods that
`SUPServer`	Command and control operations for an Unwired Server instance, for example start, stop, and ping.
`SUPCluster`	Manage cluster security, monitoring configuration and domain creation for a cluster instance, and so on.
`SUPDomain`	Manage domains, deploy packages to a domain, set security configurations for a domain, and so on.
`SUPPackage`	Deploy and configure packages by setting up subscriptions, configuring cache groups, configuring endpoint properties, and so on.
`SUPMobileBusinessObject`	View mobile business object properties, operations, errors, endpoints, and so on.
`SUPOperations`	View operation properties, errors, endpoints, and so on.
`SUPDeviceUser`	Create templates, configure device settings, manage device users, lock devices, and so on.
`SUPMonitor`	Perform monitoring functions like viewing histories, summaries, details, and performance data for various platform components, and export data as required.
`SUPServerLog`	View, filter, delete and refresh logs, configure appenders, and so on, for Unwired Server and its embedded services like replication and messaging synchronization.
`SUPDomainLog`	View, filter, delete and refresh, and so on, a domain logs instance.
`SUPServerConfiguration`	Configure an Unwired Server instance, as well as its listeners. All methods of this interface, except the apple push notification-related properties are metadata-based.

Interface	Includes methods that
`SUPSecurityConfiguration`	Create, manage, and configure a security configuration with at least one authentication provider. You can add other providers (authentication, authorization, attribution, and audit) as required.
`SUPMobileWorkflow`	Manage and configure deployed mobile workflow packages.

See also

- *Data Encryption Implementation* on page 122

Javadocs

The administration client API is installed with the `com.sybase.sup.admin.client` package. This package includes Javadocs. Use the Sybase Javadocs for your complete API reference.

As you review the contents of this document, ensure you review the reference details documented in the Javadoc delivered with this API. By default, Javadocs are installed to `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\AdminClientAPI\com.sybase.sup.adminapi\docs\api\index.html`.

The top left navigation pane lists all packages installed with Unwired Platform. The applicable documentation is available with `com.sybase.sup.admin.client` package. Click this link and navigate through the Javadoc as required.

CHAPTER 10 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 installation directories to ensure a successful installation.

Guidelines for interpreting this information on the Unwired Platform installation directories:

- The following tables document only the high-level folder structure in a complete installation on a single server.
- In all installations, most of the directories listed have subfolders.
- In custom installations, including installations for cluster environments, some of the subfolders listed will not be present.

By default Sybase Unwired Platform is installed to the `C:\Sybase\UnwiredPlatform` directory. You may have specified a different location.

Table 22. Unwired Platform 1.5.2 installation directory subfolders

Folder	Description
`_jvm`	Files for Java Virtual Machine used by uninstaller.
`Eclipse`	Files supporting the Eclipse development environment. **Note:** Present in developer installations only.
`InstallLogs`	The output location of log files created each time Unwired Platform installer is used. Use these logs to troubleshoot issues with the installer.
`JDKx.x.x_x` or `JDKx.x.x_x-x64`	Files used for version of JDK required by Unwired Platform. If the folder ends in -x64, this is the JDK for 64-bit operating systems in a production deployment environment.

Folder	Description
`scc_cert`	Certificate file used for Sybase Control Center.
`Servers`	Server components that make up Unwired Platform and its mobile middleware services.
`Servers\Advantage910`	Device management components used to administer devices from Sybase Control Center. Includes online help.
`Servers\MessagingServer`	Synchronization components used for messaging-based synchronization.
`Servers\SQLAnywhere11`	Synchronization components used for replication-based synchronization. Frequently used folders include: • `BINXX` – for utilities you might use. • `data` – for database files used by Unwired Platform.
`Servers\UnwiredServer`	The application server used in an Unwired Platform mobility environment.
`Servers\UnwiredServer\licenses`	Location where Unwired Platform licenses are saved. Every time a license is updated, copy new licenses here.
`ThirdParty`	Location where required runtime files for other components integrated into the Unwired Platform environment are saved.
`Uninstallers`	The executable and supporting files used to uninstall Unwired Platform.
`Unwired_WorkSpace`	Executables and supporting files used by Unwired Workspace. **Note:** Present in developer installations only.
`Util`	Contains utilities installer executes to check and validate external information, such as third party software installations, database information, and Windows account information.

Your Sybase Unwired Platform license includes the Sybase Control Center. By default, Sybase Control Center is installed to the `C:\Sybase\SCC-3_0` directory.

Note: If you have other Sybase Products installed, you may have two different versions of Sybase Control Center installed. Unwired Platform requires 3.0, so only this directory structure is documented.

Table 23. Sybase Control Center 3.0 installation directory subfolders

Folder	Description
`auth`	Library files used for related services in SCC. For example, JAAS.
`bin`	Scripts you can use to start or stop components of the SCC management framework.
`common`	Required files shared by SCC components.
`conf`	Configuration files used for SCC, including security providers for administration logins.
`ldap`	The LDAP related files for SCC.
`log`	Log files used by SCC and it's console plugins used capture management framework events exclusively. No Unwired Platform data is captured here, except for administration logins.
`plugins`	Location for managed resource plug-ins, including one for Unwired Platform.
`rtlilb`	Runtime library files used by SCC.
`server`	Class and library files used by the management framework server.
`services`	Class and library files used by SCC services.
`shared`	Class and library files shared by SCC and its plugins.
`utility`	Various utilities used by SCC.

Port Number Reference

Change Sybase Unwired Platform component port numbers as needed postinstallation.

Proceed with caution when changing port numbers because the change might impact other configuration files that point to that port. You need to be aware of the default Sybase Control Center port numbers so you do not accidentally use these ports when you change Sybase Unwired Platform ports. You can change some Sybase Control Center default ports, but, in some cases, you should not.

Note: To make Unwired Server port number changes, you need to temporarily stop the other service consuming those ports so that Unwired Server can start properly, make the change using Sybase Control Center, and then restart.

Port	Description	Default port	How to change postinstallation
OpenDS LDAP server	Developer Editions only. Port number on which the OpenDS LDAP server listens for requests. The default administrator password for the LDAP server is **secret**.	10389	Edit the `ds-cfg-listen-port: 10389` property in *<Unwired_Platform_InstallDir>*`\Servers\UnwiredServer\OpenDS\config\config.ldif`. **Note:** If you change the port, remember to change the edit Provider URL property.
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. See *System Administration > Environment Setup > Databases > Changing a CDB Port in a Clustered Environment*.
Management ports	IIOP port number on which the Unwired Server listens for requests	2000 2001 for secure management (disabled by default)	Configure in Sybase Control Center by expanding the Servers > *<ServerName>* folder and selecting Server Configuration. In the General tab, select the Communication Ports subtab and enter a new management port or secure management port, as required. See *Sybase Control Center online help > Configuring > Configuring Unwired Platform > Unwired Server > Server Properties > General Server Ports > Configuring Communication Port Properties*.

Port	Description	Default port	How to change postinstallation
Data Change Notification (DCN)	Port numbers on which Sybase Control Center listens for requests	8000 for HTTP 8001 for HTTPS	Configure in Sybase Control Center by expanding the Servers > *<ServerName>* folder and selecting Server Configuration. In the General tab, select the Communication Ports subtab and enter a new DCN port or secure DCN port, as required. See *Sybase Control Center online help > Configuring > Configuring Unwired Platform > Unwired Server > Server Properties > General Server Ports > Configuring Communication Port Properties*.
Replication-Based Synchronization (RBS) and Messaging-Based Synchronization (MBS)	Port numbers on which Unwired Server synchronizes data between the enterprise information system and mobile devices MBS port uses a proprietary encryption method, so MBS communication is always encrypted.	2480 for RBS 2481 for RBS (secure) 5001 for MBS	Configure in Sybase Control Center by expanding the Servers > *<ServerName>* folder and selecting Server Configuration. In the Replication or Messaging tab, select the Synchronization Listener subtab and enter a new synchronization port, as required. **Note:** If there is a port conflict for 2480 or 2481, Unwired Server cannot start. This means you cannot use SCC to modify these ports. Therefore, you need to temporarily stop the other service currently using 2480 and start Unwired Server so that you can change the required ports in SCC. For RBS, see *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Properties > Replication > Configuring General Synchronization Properties*. For MBS, see *Sybase Control Center online help > Configure > Configuring Unwired Platform > Unwired Server > Server Properties > Messaging > Configuring Messaging Synchronization Properties* .

Port	Description	Default port	How to change postinstallation
Advantage Database Server	Port number for the messaging database	6262	Port changes are not recommended.
Messaging server administration	Port number for the messaging service for Sybase Messaging Clients	5100 for administration services	Not alterable through the Sybase Control Center. Use the `<<UnwiredPlatform_InstallDir>>\Servers\Messaging Server\Bin\AdminWebServicesTool.exe` command line tool to change the messaging service Web service port. This tool has built in online help describing how to use the tool. From the command prompt run: `<UnwiredPlatform_InstallDir>\Servers\Messaging Server\Bin>AdminWebServicesTool.exe set=<port> restart`

Port	Description	Default port	How to change postinstallation
Sybase Control Center	Additional default port numbers of which to be aware, when modifying port numbers	9999 for default RMI agent port 2100 for default JMS messaging service port 3638 for default SCC repository database port 8282, 8283 for default Web container ports	• 9999 – default RMI agent port. The port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\RMI\service-config.xml` • 2100 – default JMS Messaging Service port. The port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\Messaging\service-config.xml` • 3638 – default SCC Repository database port. The default port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\SccSADataserver\service-config.xml` • 8282, 8283 – default Web Container ports. The default ports are set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\EmbeddedWebContainer\service-config.xml` Before you make any changes to these files, stop Sybase Unified Agent service. Start the service after you complete the changes. If any of the sub-systems fail to start, check the SCC `agent.log` for error messages.

Port	Description	Default port	How to change postinstallation
Relay Server	Port numbers on which Relay Server listens for requests	80 for HTTP 443 for HTTPS	Edit `<<UnwiredPlatform_InstallDir>>\Servers\UnwiredServer\config\relayserver.properties`. Run `regRelayServer.bat` after editing the file. Ensure that the Web server (IIS or Apache) is configured with these ports specified in `relayserver.properties`: • `relayserver.http_port = 80` • `relayserver.https_port = 443` See *System Administration > System Reference > Command Line Utilities > Relay Server Utilities > Register Relay Server (regRelayServer) Utility*.
Afaria Server	Port number on which Afaria Server listens for requests	4041 for HTTP 4444 for HTTPS 4343 for Afaria Database 3007 for XNET 3008 for XNETS	Modify Afaria Server default ports through the Afaria Administrator Web interface, which is launched from Sybase Control Center after you register the Afaria Server. See *System Administration > Environment Setup > Afaria Setup* for information about setting up Afaria Server in the Sybase Unwired Platform environment.

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	Advantage Database Server	The database server that manages the messaging database.	This service must be started before Sybase Messaging Service can be started.
	OpenDS	The default LDAP server for development authentication and authorization. If you are using a provider other than LDAP, you do not need to use this service. Appears in the Developer Edition installation only.	None.

Component	Service	Description	Dependencies
	Consolidated Database (the service name depends on the data tier installation type): • If the data tier is installed with Unwired Server on the same host: SybaseUnwiredPlatform<host-name>Database<install_number> • If the data tier is installed by itself: SybaseUnwiredPlatformConsolidatedDatabase	The CDB services.	The data service must be started before the Unwired Server service can be started.
	SybaseUnwiredPlatform<*cluster_ name*>Server<*install_number*>	The Unwired Server service.	Depends on the CDB service startup and relay server service (if used).
	Sybase Messaging Service	The synchronization service that facilitates communication with device client applications. Stopping this service also stops the Advantage Database service, SybaseUnwiredPlatform service, OpenDS service, and the CDB and sample database.	Depends on Advantage Database Server startup.
Afaria Server	Afaria Client Service	Appears only on desktop computers that are Afaria clients. Initiates connections with an Afaria Server to run sessions.	None.
	Afaria Database	The service required to run the Afaria Server Database. The Afaria Database service is set to automatic. But, if the service has been stopped or is not running, starting the Afaria server service does not start the Afaria Database service if it is not running. In this case, you must manually start the database.	Afaria Server

Component	Service	Description	Dependencies
	Afaria Server	The software component that manages the Afaria runtime. Starting the server does not start the database. You must start that service manually.	Afaria Database
Relay Server Outbound Enabler	There are three services, one for each service type: • SybaseUnwiredPlatform<clusterName>RSOE<install_number> • SybaseUnwiredPlatform<clusterName>IMORSOE<install_number> • SybaseUnwiredPlatform<clusterName>WebserverRSOE<install_number>	The relay-server outbound enabler (RSOE) service that enables communication from the relay server to Unwired Server. • If you use the Start menu, a shortcut, or `startrsoe.bat`, the RSOE services (RSOE, IMORSOE, and WebServerRSOE) start automatically. • If you use the Windows Services manager to start Unwired Server, the RSOE services do not start. Instead, start the RSOE services first, which automatically starts the Unwired Server service.	Depends on the Unwired Server service startup, and appears only after you run `regRelayServer.bat`.
Sybase Control Center	Sybase Unified Agent *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.

See also

- *Setting Up Data Tier Nodes* on page 54

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	OpenDS LDAP (Development Edition licenses only)	`OpenDS_service.exe`, `java.exe`
	Replication-based synchronization services (via MobiLink)	Consolidated database: `dbsrv11.exe` Synchronization: `mlsrv11.exe`
	Messaging-based synchronization services	`AdminWebServices.exe`, `ampservice.exe`, `JMSBridge.exe`, `LBManager.exe`, `OBMO.exe`, `OBServiceManager.exe` Messaging database: `ads.exe`
Relay server	Relay server	`rshost.exe`
	RSOE	`rsoe.exe`
Sybase Control Center	Sybase Unified Agent	`uaservice.exe`
	Sybase Control Center repository database	`dbsrv11.exe`

Security Provider Configuration Properties

Security providers implement different properties, depending on whether or not they support authentication, authorization, audit, or attribution.

Platform administrators can configure application security properties in the Sybase Control Center for Unwired Platform console. These properties are then transcribed to the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\CSI\default.xml` file. A new section is created for each provider you add.

Security provider configuration files typically store data in a key=value structure, which means each item of data (the key) has a name, and its value is its contents.

LDAP Configuration Properties

Use these properties in your `csi.properties` file to control your LDAP service. Use these properties to configure the LDAP provider used to authenticate SCC administration logins or to configure the LDAP provider used to authenticate device application logins. If you are creating a provider for device application logins, then Unwired Platform administrators use Sybase Control Center to write these properties to the

`<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\CSI\default.xml` file.

Unwired Server implements a Java LDAP provider through a common security interface used by other Sybase products like Sybase Control Center.

The Java LDAP provider consists of three provider modules, each of which is in the `com.sybase.security.ldap` Java package. This is why the syntax used between Sybase Control Center provider and Unwired Server varies.

- The LDAPLoginModule class provides the authentication services.
- LDAP role-based authorization is accomplished using the core RoleCheckAuthorizer or LDAPAuthorizer. LDAP-specific authorization provider is not necessary if LDAP authentication is employed. One can use LDAPAuthorizer for LDAP role-based authorization when non-LDAP authentication is used.
- The LDAPAttributer class provides attribution services.

When configuring modules or general server properties in Sybase Control Center, note that properties and values can vary, depending on which module or server type you configure.

Property	Default Value	Description
ServerType	None	Optional. The type of LDAP server you are connecting to: • `sunone5` -- SunOne 5.x OR iPlanet 5.x • `msad2k` -- Microsoft ActiveDirectory, Windows 2000 • `nsds4` -- Netscape Directory Server 4.x • `openldap` -- OpenLDAP Directory Server 2.x The value you choose establishes default values for these other authentication properties: • RoleFilter • UserRoleMembership • RoleMemberAttributes • AuthenticationFilter • DigestMD5Authentication • UseUserAccountControl

Property	Default Value	Description
ProviderURL	`ldap://local-host:389`	The URL used to connect to the LDAP server. Without this URL configured, Unwired Server cannot contact your server. Use the default value if the server is: • Located on the same machine as your product that is enabled with the common security infrastructure. • Configured to use the default port (389). Note that Development Editions of Unwired Platform include an OpenDS LDAP server that runs on a nonstandard port of 10389. However, most LDAP servers use the standard port of 389. Otherwise, use this syntax for setting the value: `ldap://<hostname>:<port>`
DefaultSearchBase	None	The LDAP search base that is used if no other search base is specified for authentication, roles, attribution and self registration: 1. `dc=<domainname>,dc=<tld>` For example, a machine in sybase.com domain would have a search base of dc=sybase,dc=com. 2. `o=<company name>,c=<country code>` For example, this might be o=Sybase,c=us for a machine within the Sybase organization.
SecurityProtocol	None	The protocol to be used when connecting to the LDAP server. To use an encrypted protocol, use "ssl" instead "ldaps" in the url. **Note:** ActiveDirectory requires the SSL protocol when setting the value for the password attribute. This occurs when creating a user or updating the password of an existing user.

Property	Default Value	Description
AuthenticationMethod	simple	The authentication method to use for all authentication requests into LDAP. Legal values are generally the same as those of the java.naming.security.authentication JNDI property. Choose one of: • simple — For clear-text password authentication. • DIGEST-MD5 — For more secure hashed password authentication. This method requires that the server use plain text password storage and only works with JRE 1.4 or later. See the*Java Sun* Web site for more information.
AuthenticationFilter	For most LDAP servers: `(&(uid={uid})(objectclass=person))` or For Active Directory email lookups: `(&(userPrincipalName={uid})(objectclass=user))[ActiveDirectory]` For Active Directory Windows username lookups: `(&(sAMAccountName={uid})(objectclass=user))`	The filter to use when looking up the user. When performing a username based lookup, this filter is used to determine the LDAP entry that matches the supplied username. The string "{uid}" in the filter is replaced with the supplied username.

Property	Default Value	Description
AuthenticationScope	onelevel	The authentication search scope. The supported values for this are: • `onelevel` • `subtree` If you do not specify a value or if you specify an invalid value, the default value is used.
AuthenticationSearchBase	none	The search base used to authenticate users. If this value is not specified, the LDAP DefaultSearchBase is used.
BindDN	none	The user DN to bind against when building the initial LDAP connection. In many cases, this user may need read permissions on all user records. If you do not set a value, anonymous binding is used. Anonymous binding works on most servers without additional configuration. However, the LDAP attributer may also use this DN to create the users in the LDAP server. When the self-registration feature is used, this user may also need the requisite permissions to create a user record. This behavior can occur if you do not set useUserCredentialsToBind to `true`. In this case, the LDAP attributer uses this DN to update the user attributes.

Property	Default Value	Description
BindPassword	none	BindPassword is the password for BindDN, which is used to authenticate any user. BindDN and BindPassword are used to separate the LDAP connection into units. The AuthenticationMethod property determines the bind method used for this initial connection. If you use an encrypted the password using the CSI encryption utility, append .e to the property name. For example: `CSI.loginModule.7.options.BindPassword.e=1-AAAAEgQQOLL+LpXJO8fO9T4SrQYRC9lRT1w5ePfdczQTDsP8iACk9mDAbm3F3p5a1wXWKK8+NdJuknc7w2nw5aGJlyG3xQ==` For details on the CSI encryption utility see *System Administration > System Reference > Command Line Utilities*.
RoleSearchBase	none	The search base used to retrieve lists of roles. If this value is not specified, the LDAP DefaultSearchBase is used.
RoleFilter	For SunONE/iPlanet: `(&(objectclass=ldapsubentry) (objectclass=nsroledefinition))` For Netscape Directory Server: (`objectclass=groupofnames) (objectclass=groupofuniquenames`s)) For ActiveDirectory: `(objectclass=groupofnames) (objectclass=group))`	The role search filter. This filter should, when combined with the role search base and role scope, return a complete list of roles within the LDAP server. There are several default values depending on the chosen server type. If the server type is not chosen or this property is not initialized, no roles are available.

Property	Default Value	Description
RoleMemberAttributes	For Netscape Directory Server: member,uniquemember	The role's member attributes defines a comma-delimited list of attributes that roles may have that define a list of DN's of people who are in the role. These values are cross referenced with the active user to determine the user's role list. One example of the use of this property is when using LDAP groups as placeholders for roles. This property only has a default value when the Netscape server type is chosen.
RoleNameAttribute	cn	The attribute for retrieved roles that is the common name of the role. If this value is "dn" it is interpreted specially as the entire dn of the role as the role name.
RoleScope	onelevel	The role search scope. The supported values for this are: • `onelevel` • `subtree` If you do not specify a value or if you specify an invalid value, the default value is used.
UserRoleMembershipAttributes	For iPlanet/SunONE: nsRoleDN For ActiveDirectory: memberOf For all others: none	The user's role membership attributes property is used to define an attribute that a user has that contains the DN's of all of the roles as user is a member of. These comma-delimited values are then cross-referenced with the roles retrieved in the role search base and search filter to come up with a list of user's roles.
UserFreeformRoleMembershipAttributes	None	The "freeform" role membership attribute list. Users who have attributes in this comma-delimited list are automatically granted access to roles whose names are equal to the attribute value. For example, if the value of this property is "department" and user's LDAP record has the following values for the department attribute, { "sales", "consulting" }, then the user will be granted roles whose names are "sales" and "consulting".

Property	Default Value	Description
Referral	ignore	The behavior when a referral is encountered. The valid values are those dictated by LdapContext, for example, "follow", "ignore", "throw".
DigestMD5Authentication-Format	DN For OpenLDAP: User-name	The DIGEST-MD5 bind authentication identity format.
UseUserAccountContro-lAttribute	For most LDAP servers: false For ActiveDirectory: true	The UserAccountControl attribute to be used for detecting disabled user accounts, account expira-tions, password expirations and so on. ActiveDir-ectory also uses this attribute to store the above information.
controlFlag	optional	Indicates whether authentication with this login module is sufficient to allow the user to log in, or whether the user must also be authenticated with another login module. Rarely set to anything other than "sufficient" for any login module. **Note:** controlFlag is a generic login module op-tion rather than an LDAP configuration property.

See also

- *Authentication* on page 104
- *LDAP Security Provider* on page 108
- *Active Directory Considerations* on page 108

NTProxy Configuration Properties

Configure these properties to allow the operating system's security mechanisms to validate user credentials using NTProxy (Windows Native OS). Access these properties from the Authentication tab of the Security node in Sybase Control Center.

Table 24. Authentication properties

Properties	Default Value	Description
Extract Domain From Username	true	If set to true, the user name can contain the domain in the form of *<username>@<domain>*. If set to false, the default domain (described below) is always used, and the supplied user name is sent to through SSPI untouched.
Default Domain	The domain for the host computer of the Java Virtual Machine.	Specifies the default host name, if not overridden by the a specific user name domain.
Default Authentication Server	The authentication server for the host computer of the Java Virtual Machine.	The default authentication server from which group memberships are extracted. This can be automatically determined from the local machine environment, but this property to bypass the detection step.
useFirstPass	false	If set to true, the login module attempts to retrieve only the user name and password from the shared context. It never calls the callback handler.
tryFirstPass	false	If set to true, the login module first attempts to retrieve the user name and password from the shared context before attempting the callback handler.
clearPass	false	If set to true, the login module clears the user name and password in the shared context when calling either commit or abort.
storePass	false	If set to true, the login module stores the user name and password in the shared context after successfully authenticating.

See also

- *Authentication* on page 104
- *NTProxy Security Provider* on page 109
- *Setting Up the NTProxy Provider* on page 115

Domino Configuration Properties

Configure Domino properties from the Authentication and Attribution tabs in the Security node of Sybase Control Center.

Because Domino provides both authentication and attribution services, it supports these types of configuration properties:

Table 25. Authentication and attribution properties

Property	Default value	Description
Server Name	localhost	The Domino Server host name.
Server Port	63148	The Domino Server DIIOP port number.
Database File Names	none	A comma-separated list of database file names in Domino from which user roles are retrieved. The Database ACL entries are examined to retrieve the roles of the authenticated user. If this configuration option is not defined, then the roles are collected from all the available databases in the Domino Server. However, this may result in a long authentication times.
User Attributes	none	A comma-separated list of user attributes that should be retrieved from Domino Server in addition to the basic attributes the provider retrieves by default.

Table 26. Authentication properties

Property	Default Value	Description
useFirstPass	false	If set to true, the login module attempts to retrieve only the user name and password from the shared context. It never calls the callback handler.

Property	Default Value	Description
tryFirstPass	false	If set to true, the login module first attempts to retrieve the user name and password from the shared context before attempting the callback handler.
clearPass	false	If set to true, the login module clears the user name and password in the shared context when calling either commit or abort.
storePass	false	If set to true, the login module stores the user name and password in the shared context after successfully authenticating.

See also

- *Authentication* on page 104
- *Domino Security Provider* on page 110

Remedy Configuration Properties

Configure the RemedyLoginModule class to provide authentication services. Access Remedy properties from the Authentication tab in the Security node of Sybase Control Center.

Table 27. Authentication properties

Property	Default value	Description
Server Name	localhost	The Remedy AR system server host name.
Server Port	0 A value of 0 indicates that the default Remedy server port number will be used.	The Remedy AR system server port.
User Locale	None An empty value field indicates that the default local system locale will be used.	The Remedy AR system user locale.

Property	Default value	Description
Allow Guest	false	Indicates whether Remedy AR system allows a user as a guest: • true – an authenticated user can access Remedy AR. • false – an authenticated user receives error messages and cannot access Remedy AR.
Server Remote Procedure Call Port	0	The Remedy server RPC program number.
useFirstPass	false	If set to true, the login module attempts to retrieve only the user name and password from the shared context. It never calls the callback handler.
tryFirstPass	false	If set to true, the login module first attempts to retrieve the user name and password from the shared context before attempting the callback handler.
clearPass	false	If set to true, the login module clears the user name and password in the shared context when calling either commit or abort.
storePass	false	If set to true, the login module stores the user name and password in the shared context after successfully authenticating.

See also

- *Authentication* on page 104
- *Remedy Security Provider* on page 110

RADIUS Configuration Properties

For Unwired Server security, you configure properties from the corresponding tab for **Security Configuration** in Sybase Control Center. However, you can manually configure a RADIUS provider , by creating a provider defintion and adding the property name and value to the `<UnwiredServer-install-directory>\tomcat\conf\CSI\default.xml` file.

Because RADIUS is an authentication provider, it supports the following authentication-related configuration properties:

Property	Default Value	Description
Authentication Method	PAP	The authentication method to use. Valid values are *PAP* and *CHAP*.
Shared Secret	none	The secret shared between the RADIUS server and the host where the login module is executed.
Radius Server Host Name	none	The name of the host for the RADIUS server.
Radius Server Authentication Port	1812	The authentication port for the RADIUS server.
Case Sensitive Matching	false	Specifies whether or not case sensitive matching should be used when matching RADIUS server error messages and the regular expressions you configure.
Max Challenges	3	Specifies the maximum number of times the RADIUS server prompts the user for additional information.
useFirstPass	false	If set to true, the login module only attempts to retrieve the user name and password from the shared context. It never calls the callback handler.
tryFirstPass	false	If set to true, the login module first attempts to retrieve the user name and password from the shared context before attempting the callback handler.
clearPass	false	If set to true, the login module clears the user name and password in the shared context when calling either commit or abort.

Property	Default Value	Description
storePass	false	If set to true, the login module stores the user name and password in the shared context after successfully authenticating.

See also

- *Authentication* on page 104
- *RADIUS Security Provider* on page 110

EIS Data Source Connection Properties Reference

Name and configure connection properties when you create connection pools in Sybase Control Center to enterprise information systems (EIS) .

See also

- *EIS Connections* on page 80
- *EIS Connection Management Overview* on page 146

JDBC Properties

Configure Java Database Connectivity (JDBC) connection properties.

This list of properties can be used by all datasource types. Sybase does not document native properties used only by a single driver. However, you can also use native driver properties, naming them using this syntax:

```
<driver_type>:<NativeConnPropName>=<SupportedValue>
```

Note: If Unwired Server is connecting to a database with a JDBC driver, ensure you have copied required JAR files to correct locations. See *System Administration > Environment Setup > EIS Connections > Preparing Unwired Server to Connect to JDBC Databases*.

Name	Description	Supported values
After Insert	Changes the value to `into` if a database requires `insert into` rather than the abbreviated `into`.	`into`

Name	Description	Supported values
Alias For	Sets an alias name for a datasource. The alias is used for binding the same datasource with a different name, so that multiple applications can use a single datasource. **Note:** When you set a datasource name, Unwired Server ignores all other properties for this datasource. The real connection properties are extracted from the datasource definition that the aliasFor value points to.	`<name>`
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 `databaseStartCommand`, and `databaseStopCommand`. Otherwise, do not set this property, or use a value of 0 to disable the echo.

Name	Description	Supported values
Database Create Command	Specifies the operating system command used to create the database for this datasource. If this command is defined and the file referenced by ${databaseFile} does not exist, the command is run to create the database when an application component attempts to obtain the first connection from the connection pool for this datasource.	`<command>` Example: `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32\dbinit -q ${databaseFile}`
Database File	Indicates the database file to load when connecting to a datasource. Use this property when the path to the database file differs from the one normally used by the database server. If the database you want to connect to is already running, use the databaseName connection parameter.	`<string>` Supply a complete path and file name. The database file you specify must be on the same host as the server.

Name	Description	Supported values
Database Name	Identifies a loaded database with which to establish a connection, when connecting to a datasource. Set a database name, so you can refer to the database by name in other property definitions for a datasource. If the database to connect to is not already running, use the databaseFile connection parameter so the database can be started. **Note:** For Unwired Server, you typically do not need to use this property. Usually, when you start a database on a server, the database is assigned a name. The mechanism by which this occurs varies. An administrator can use the DBN option to set a unique name, or the server may use the base of the file name with the extension and path removed.	`[DBN|default]` If you set this property to default, the name is obtained from the DBN option set by the database administrator. If no value is used, the database name is inherited from the database type.
Database Start Command	Specifies the operating system command used to start the database for this datasource. If this command is defined and the database is not running, the command is run to start the database when the datasource is activated.	`<command>` Example: `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32\dbsrv11.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\dbsrv11.exe`

Database Type	Specifies the database type.	`<database type>`

Name	Description	Supported values
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.
Max Wait Time	Sets the maximum number of seconds to wait for a connection before the request is cancelled.	`<int>` The default is 60.
Max Statements	Specifies the maximum number of JDBC prepared statements that can be cached for each connection by the JDBC driver. The value of this property is specific to each JDBC driver.	`<int>` A value of 0 (default) sets no limit to the maximum statements.
Min Pool Size	Sets the minimum number of connections allocated to the pool for this datasource.	`<int>` A value of 0 (default) sets no limit to the minimum connection pool size.

Name	Description	Supported values
Network Protocol	Sets the protocol used for network communication with the datasource. Use this property (along with the driverClass, and dataSourceClass properties) only if you do not have a predefined database-type entry in Unwired Server for the kind of SQL database you are connecting to. For example, you may be required to use this property for MySQL database connections.	The network protocol is JDBC driver vendor-specific. There are no predefined values. See the driver vendor's JDBC documentation.
Password	Specifies the password for connecting to the database.	`[default|<password>]`
Ping and Set Session Auth	Runs the ping and session-authorization commands in a single command batch; may improve performance. You can only enable the Ping and Set Session Auth property if you have enabled the Set Session Auth property so database work runs under the effective user ID of the client.	`[True|False]` Set to true to enable, or false to disable.
Ping Connections	Pings connections before attempting to reuse them from the connection pool.	`[True|False]` Set to true to enable ping connections, or false to disable.
Ping SQL	Specify the SQL statement to use when testing the database connection with ping.	`[default|<statement>]` Replace <statement> with the SQL statement identifier. The default is "select 1".
Port Number	Sets the server port number where the database server listens for connection requests.	`[default|<port>]` Replace <port> with the TCP/IP port number to use (that is, 1 – 65535). If you set the value as default, the default protocol of the datasource is used.

Name	Description	Supported values
PS Maximum Blob Length	Indicates the maximum number of bytes allowed when updating a BLOB datatype using PreparedStatement.setBytes.	`[default|<int>]` Replace <int> with the number of bytes allowed during an update. The default is 16384.
PS Maximum Clob Length	Indicates the maximum number of characters allowed when updating a CLOB datatype using PreparedStatement.setString.	`[default|<int>]` Replace <int> with the number of bytes allowed during an update. The default is 16384.
Role Name	Sets the database role that the user must have to log in to the database.	`[default|<name>]` If you set this value to default, the default database role name of the datasource is used.
Server Name	Defines the host where the database server is running.	`<name>` Replace <name> with an appropriate name for the server.
Service Name	Defines the service name for the datasource. For SQL Anywhere servers, use this property to specify the database you are attaching to.	`<name>` Replace <name> with an appropriate name for the service.
Set Session Auth	Establishes an effective database identity that matches the current mobile application user. If you use this property, you must also use setSessionAuthSystemID to set the session ID. Alternately you can pingAndSetSessionAuth if you are using this property with pingConnection. The pingAndSetSessionAuth property runs the ping and session-authorization commands in a single command batch, which may improve performance.	`[true|false]` Choose a value of 1 to use an ANSI SQL set session authorization command at the start of each database transaction. Set to 0 to use session-based authorizations.

Name	Description	Supported values
Set Session Auth System ID	If Set Session Authorization is enabled, specifies the database identity to use when the application server accesses the database from a transaction that runs with "system" identity.	`<database identity>` Replace <database identity> with the database identifier.
Start Wait	Sets the wait time (in seconds) before a connection problem is reported. If the start command completes successfully within this time period, no exceptions are reported in the server log. startWait time is used only with the databaseStartCommand property.	`<int>` Replace <int> with the number of seconds Unwired Server waits before reporting an error.
Truncate Nanoseconds	Sets a divisor/multiplier that is used to round the nanoseconds value in a java.sql.Timestamp to a granularity that the DBMS supports.	`[default|<int>]` The default is 10 000 000.
Use Quoted Identifiers	Specifies whether or not SQL identifiers are quoted.	`[True|False]` Set to true to enable use of quoted identifiers, or false to disable.
User	Identifies the user who is connecting to the database.	`[default|<user name>]` Replace <user name> with the database user name.
XA Datasource Class	Specifies the class name or library name used to support two-phase commit transactions, and the name of the XA resource library.	`<class name>` Replace <class name> with the class or library name. • SQL Anywhere database: `com.sybase.jdbc3.jdbc.SybXADataSource` • Oracle database: `oracle.jdbc.xa.client.OracleXADataSource`

SAP Java Connector Properties

Configure SAP Java Connector (JCo) connection properties.

For a comprehensive list of SAP JCo properties you can use, see *http://help.sap.com/javadocs/NW04/current/jc/com/sap/mw/jco/JCO.html*.

Note: If Unwired Server is connecting to SAP with a Java connector, ensure you have copied required files to correct locations. See *System Administration > Environment Setup > EIS Connections > Preparing Unwired Server to Connect to SAP*.

Table 28. General connection parameters

Name	Description	Supported values
Client Number	Specifies the SAP client.	Three-digit client number; preserve leading zeros if they appear in the number
Logon User	Specifies the login user ID.	User name for logging in to the SAP system
Password	Specifies the login password.	Password for logging in to the SAP system
Language	Specifies a login language.	ISO two-character language code (for example, EN, DE, FR), or SAP-specific single-character language code.
System Number	Indicates the SAP system number.	SAP system number
Host Name	Identifies the SAP application server.	Host name of a specific SAP application server
Message Server	Identifies the SAP message server.	Host name of the message server
Gateway Host	Identifies the SAP gateway host.	Host name of the SAP gateway Example: `GWHOST=hs0311`
Gateway Service	Identifies the SAP gateway service.	Service name of the SAP gateway Example: `GWSERV=sapgw53`
R/3 Name	Specifies R/3 name.	Name of the SAP system
Server Group	Identifies the group of SAP application servers.	Group name of the application servers

Name	Description	Supported values
External Server Program	Identifies the program ID of the external server program.	Path and name of the external RFC server program, or program ID of a registered RFC server program Example: `TPNAME=/sap/srfcserv`
External Server Program Host	Identifies the host of the external server program. This information determines whether the RFC client connects to an RFC server started by the SAP gateway or to an already registered RFC server. **Note:** If the gateway host and external server program host are different, make sure that the SAP gateway has access to start the server program through a remote shell.	Host name of the external RFC server program Example: `TPHOST=hs0311`
Remote Host Type	Identifies the type of remote host.	2: R/2 3: R/3 E: external
RFC Trace	Specifies whether or not to enable RFC trace.	0: disable 1: enable
Initial Codepage	Identifies the initial code page in SAP notation. A code page is used whenever character data is processed on the application server, appears on the front end, or is rendered by a printer.	Four-digit SAP code page number

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.	0: no debugging 1: attach a visible SAPGUI and break at the first ABAP statement of the invoked function module
Remote GUI	Specifies whether a remote SAP graphical user interface (SAPGUI) should be attached to the connection. Some older BAPIs need an SAPGUI because they try to send screen output to the client while executing.	0: no SAPGUI 1: attach an "invisible" SAPGUI, which receives and ignores the screen output 2: attach a visible SAPGUI For values other than 0 a SAPGUI needs to be installed on the machine, where the client program is running. This can be either a Windows SAPGUI or a Java GUI on Linux/Unix systems.
Get SSO Ticket	Generates an SSO2 ticket for the user after login to allow single sign-on. If RfcOpenConnection() succeeds, you can retrieve the ticket with RfcGetPartnerSSOTicket() and use it for additional logins to systems supporting the same user base.	0: do not generate SSO2 ticket 1: generate SSO2 ticket
Use Cookie Version 2	Indicates whether or not to use the specified SAP Cookie Version 2 as the login ticket instead of user ID and password.	User: $MYSAPSSO2$ Password: Base64-encoded ticket Login with single sign-on is based on secure network connection (SNC) encryption and can only be used in combination with an SNC.

Name	Description	Supported values
Use X509	Indicates whether or not to use the specified X509 certificate as the log-in certificate instead of user ID and password.	User: $X509CERT$ Password: Base64-encoded ticket Login with X509 is based on secure network connection (SNC) encryption and can only be used in combination with an SNC.
Logon Check	Enables or disables login check at open time.	0: disable 1: enable If you set this to 0, RfcOpenConnection() opens a network connection, but does not perform the login procedure. Therefore, no user session is created inside the back-end system. This parameter is intended only for executing the function module RFC_PING.
Additional GUI Data	Provides additional data for graphical user interface (GUI) to specify the SAProuter connection data for the SAPGUI when it is used with RFC.	/H/ *`router string`* : the entire router string for the SAPGUI /P/ *`password`* : specify this value if the password for the SAPGUI connection is not the same as the password for the RFC connection.
GUI Redirect Host	Identifies which host to redirect the remote graphical user interface to.	Host name
GUI Redirect Service	Identifies which service to redirect the remote graphical user interface to.	Name of the service
Remote GUI Start Program	Indicates the program ID of the server that starts the remote graphical user interface.	Program ID of the server
SNC Mode	Enables or disables secure network connection mode.	0: off 1: on
SNC Partner	Identifies the secure network connection partner.	Secure network connection name of the application server (for example, p:CN=R3, O=XYZ-INC, C=EN)

Name	Description	Supported values
SNC Level	Specifies the secure network connection security level.	1: digital signature 2: digital signature and encryption 3: digital signature, encryption, and user authentication 8: default value defined by backend system 9: maximum value that the current security product supports
SNC Name	Indicates the secure network connection name. This property overrides the default secure network connection partner.	Token or identifier representing the external RFC program
SNC Service Lib Path	Idenitifies the path to the library that provides secure network connection service.	Full path and name of third-party security library
R/2 Destination	Identifies a configured R/2 system defined in the sideinfo configuration.	
Logon ID	Defines the string for SAPLOGON on 32-bit Windows.	String key to read parameters from the saplogon.ini file created by the SAPLogon GUI program on Windows
External Authentication Data	Provides data for external authentication (PAS). This is an old login mechanism similar to SSO; Sybase recommends that you do not use this approach.	
External Authentication	Specifies type of external authentication (PAS). See External Authentication Data property.	

SAP DOE-C Properties

Configure SAP Data Orchestration Engine Connector (DOE-C) properties. This type of connection is available in the list of connection templates only when you deploy a DOE-C package. No template exists for these types of connections.

Name	Description	Supported values
SAP User Login	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 must be set to perform actions on subscriptions.	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 7 minutes.

Name	Description	Supported values
DOE Packet Drop Size	Specifies the size 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. Messages larger than the packet drop threshold size causes the subscription to enter the DOE packet drop state and become unusable.	Positive value (in bytes). The default is 1MB.
Service Address	Specifies the DOE URL.	Valid DOE URL.

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

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 consolidated 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

The relay server uses three command line utilities: rshost, regRelayServer, and rsoeservice.

Launch these utilities from the command line; they are not available from any other administration tool.

See also

- *Relay Server Setup* on page 62
- *Relay Server Documentation* on page 63
- *Relay Server Configuration Files* on page 336
- *Configuring Relay Server Outbound Enabler Logging* on page 232

Relay Server Host (rshost) Utility

Maintains relay server state information across client requests and RSOE sessions and manages the log file used by the relay server. This utility is located in *UnwiredPlatform_InstallDir*`\Servers\SQLAnywhere11\Mobilink\relayserver\IIS\Bin32\IAS_relay_server\Server`.

Sybase recommends that you configure the Relay Server host as a Windows service. For information, see *Sybase Unwired Platform System Administration Guide > Environment Setup > Relay Server Setup > Configuring Relay Server to Run as a Windows Service*.

Syntax

Variable declaration:

```
rshost [option]+
```

Parameters

- <***option***> – the following options can be used to configure `rshost`. They are all optional.

Option	Description
`-f <filename>`	File name of the relay server configuration file.
`-o <filename>`	File name to use for logging.
`-oq`	Prevent popup window on startup error.
`-q`	Run in minimized window.
`-qc`	Close window on completion.

Option	Description
`-u`	Update configuration of a running relay server.
`-ua`	Archive the log file to <yymmdd><nn>.olg and truncate.

Examples

- **Example 1: Updating the relay server configuration –** The following command updates rshost with changes made to relay server configuration:

  ```
  rshost –u –f rs.config
  ```

- **Example 2: Setting configuration and log files for rshost –** The following command specifies the configuration and log files for the relay server host; it configures the `rshost` to run in a minimized window and close the window on completion:

  ```
  rshost -q -qc -f C:\Sybase\UnwiredPlatform\Servers
  \SQLAnywhere11\MobiLink\relayserver\IIS
  \Bin32\IAS_relay_server\Server\rs.config -o c:\Sybase\log
  \rs.log
  ```

  For more information on the relay server host utility, see *rshost utility* in the SQL Anywhere documentation.

Register Relay Server (regRelayServer) Utility

Loads the `relayserver.properties` information into the cluster and registers the relay server. Run this script any time you make changes to the `relayserver.properties` file. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin`.

Prerequisites: You need JDK 1.6.0_16 or higher to use this utility. 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.

Before registering the relay server, update the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\config\relayserver.properties` file to reflect the correct relay server details. Make sure the cluster database is running before you execute the command.

Syntax

```
regRelayServer [install auto | install manual | remove]
```

Parameters

- **install auto –** configures installation settings to register the relay server as a Windows service in auto-start mode.
- **install manual –** configures installation settings to register the relay server as a Windows service in manual start mode.
- **remove –** unregister the relay server and removes the RSOE services.

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
  ```

See also

- *Configuring Relay Server to Run as a Windows Service* on page 69

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.

Certificate Creation (createcert) Utility

Generates X.509 certificates or signs pregenerated certificate requests. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32`.

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` or `-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`
- `Enter file path of signer's certificate` – (optional) 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.
- `Serial number` – (optional) 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 that 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. By default, certificates are not certificate authorities (n).
- `Key usage` – supply a comma-separated list of numbers that indicate how the certificate's private key can be used. The default, which depends on whether the certificate is a certificate authorty or not, 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 PCKS10 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: Creating a signed certificate –** This example creates a signed certificate. No file name is provided for the signer's certificate, which makes it a self-signed root certificate.

```
<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers
\SQLAnywhere11\BIN32>createcert
SQL Anywhere X.509 Certificate Generator Version 11.0.1.2405
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
State/Province: CA
Locality: Dublin
Organization: MyCompany
Organizational Unit: PTO
Common Name: MyCompany
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), a self-signed root certificate should be created 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
```

See also

- *Creating a Certificate Authority* on page 93
- *Generating a Certificate Request* on page 94
- *Generating an Afaria Certificate* on page 103
- *Sharing an Unwired Server Certificate with Afaria Server* on page 103

Key Creation (createkey) Utility

Creates RSA and ECC key pairs for use with Unwired Server end-to-end encryption. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32`.

Syntax

```
createkey
```

When you run **createkey**, you are prompted for the following 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

- **Basic Example –** This 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
```

See also

- *Configuring Unwired Server Administration Certificates* on page 90

Key Tool (keytool) Utility

The key tool is a Java development kit utility that allows you to manage a keystore (database) of private keys and their associated X.509 certificates. The keytool utility also manages certificates from trusted entities. This utility is located in

<*UnwiredPlatform_InstallDir*>\<*UnwiredPlatform*>\Servers\SQLAnywhere11\Sun\JRE160_x86\bin.

keytool enables users to create and manage their own public/private key pairs and associated certificates for use in self-authentication or data integrity and authentication services, using digital signatures. It also allows users to cache the public keys (in the form of certificates) of their communicating peers.

Syntax

```
keytool -list | -printcert | -import | -export| -delete | -selfcert |
-certreq | -genkey [options]
```

Parameters

- **-list –** displays the contents of a keystore or keystore entry.
- **-printcert –** displays the contents of a certificate stored in a file. Check this information before importing a certificate as a trusted certificate. Make sure certificate fingerprints as as expected.
- **-import –** imports a certificate to:

  1. Add a certificate or certificate chain to the list of trusted certificates, or
  2. Import 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*>.
- **[*options*] – these options are available with the -genkey parameter:**

Option	Description
`-keystore <keyStoreLocation>`	The 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, that 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 that is used to protect the integrity of the keystore. The password must be at least 6 characters long. It must be 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 used to protect the private key of the key pair. If you press enter at the prompt, the key password is set to the same password as for the keystore. `keypass` must be at least 6 characters long.

Examples

- **Example 1: Viewing the keystore –** Display the contents of the keystore:

  ```
  keytool -list -keystore <filePath>\keystore.jks -storepass
  <storepass>
  ```

- **Example 2: Importing a certificate reply from a CA –** Import a certificate:

  ```
  keytool -import -file <certificate file> -keystore
  <filePath>\keystore.jks -storepass <storepass> -noprompt -
  trustcacerts -alias <alias>
  ```

- **Example 3: Deleting a certificate –** Delete a certificate:

```
keytool -delete -alias <alias> -keystore <filePath>
\keystore.jks -storepass <storepass>
```

- **Example 4: Generating a key pair –** Create a new key entry in a keystore:

  ```
  keytool -genkey -keystore <filePath>\keystore.jks
  ```

  The certificate request must be signed by a CA. Alternatively, you can self-sign the certificate 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.

Unwired Server Service (sup-server-service) Utility

Installs, removes, starts, and stops Unwired Server as a Windows service from the command line. This utility is located in *<UnwiredPlatform_InstallDir>*\Servers\UnwiredServer\bin.

Prerequisites: You need JDK 1.6.0_16 or higher to use this utility. 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

```
sup-server-service [install auto | install manual | remove | start |
stop]
```

Parameters

- **install auto –** installs Unwired Server as a Windows service in auto-start mode.
- **install manual –** installs Unwired Server as a Windows service in manual start mode.
- **remove –** uninstalls the Unwired Server service.
- **start –** starts the Unwired Server service.
- **stop –** stops the Unwired Server service.

Examples

- **–** This command installs Unwired Server as an auto-start Windows service:

  ```
  sup-server-service   install auto
  ```

Usage

Use the command line utility only when you cannot start or stop the service using the Unwired Server desktop shortcuts.

Runtime Configuration (configure-mms) Utility

Applies manual Unwired Server configurations. Use the `configure-mms.bat` utility before restarting Unwired Server. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin`.

Syntax

```
configure-mms [<clusterName> <installType> <cdbHostName> <cdbPort>
<cdbUser> <cdbPassword> <cdbServerName>]
```

Parameters

- **clusterName –** the name of the cluster to which the Unwired Server belongs.
- **installType –** the type of consolidated database (CDB) installation: default or remote.
- **cdbHost –** the name of the machine where the existing database server is running.
- **cdbPort –** the port over which CDB communication takes place. Default: 5200.
- **cdbServerUserName –** the CDB user name. Default: dba.
- **cdbServerPwd –** the CDB user password. Default: sql.
- **cdbServerName –** the name of the database server used to manage requests for the CDB. Default: <MyComputer>_primary. If the database is on another host or is part of a cluster, you may need to use another host name.
- **cdbServiceStartMode –** the start mode of the CDB service: auto or manual.

Usage

In certain administration scenarios, manual configuration changes are required; for example, if you set a custom CDB port by manually editing the `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer\sup.properties` file. In situations that require server configuration changes to be performed outside of Sybase Control Center, use the `configure-mms.bat` utility to commit these changes.

License Upgrade (license) Utility

Upgrades the license in a served model when you purchase more licenses from Sybase. In an unserved model, you simply replace the license file. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin`.

The `license.bat` utility upgrades only Unwired Platform licenses. You cannot use it to upgrade Afaria licenses, which can be upgraded only with the Afaria Administrator.

Syntax

```
license.bat <PE> <LT> [licenseNumber]
```

Parameters

- **PE –** use the corresponding abbreviation that matches the product edition in your license. Valid product editions include:
  - EE for Enterprise Edition
  - AE for Advanced Deployment Edition
  - SK for Starter Deployment Edition
  - ED for Enterprise Developer Edition
  - PD for Personal Developer Edition
- **LT –** use the corresponding abbreviation that matches the license type in your license. Valid license types include:
  - AC for Application deployment CPU license
  - AS for Application deployment seat license
  - CP for CPU license
  - ST for Seat license
  - OT for Other license
  - SS for Standalone seat license
  - DT for Development and test license

Examples

- **Basic Example –** If your license includes an uncommented line that reads `INCREMENT SUP_BASESRVR SYBASE 2010.03316 31-mar-2010 uncounted \ VENDOR_STRING=PE=EE;LT=CP HOSTID=ANY ISSUER="CO=Sybase, \`, use this command.

  ```
  license.bat EE CP <MyLicenseNumber>
  ```

Usage

Depending on the product edition you have, the license type varies according to these guidelines:

- If you entered EE for product edition, the license type can be AC, AS, CP, or ST.
- If you entered PD for product edition, the license type can be SS.
- If you entered ED for product edition, the license type can be DT.

See also

- *Updating and Upgrading Unwired Platform Licenses* on page 247
- *Locating Information in a License File* on page 248
- *Upgrading Afaria Licenses* on page 250

Synchronization Monitor (mlmon) Utility

Monitors the progress of replication-based synchronization (RBS) and checks for potential data contention or bottlenecks. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32`.

Prerequisites: You need JDK 1.6.0_16 or higher to use this utility. If you choose to use an existing JDK option during installation, also ensure that you have set the `JAVA_HOME` environment variable to the installation location of this JDK version. Otherwise, you must use a text editor to modify the existing JDK path in the batch file itself.

Syntax

```
mlmon [connect-options|inputfile.{mlm|csv}]
```

When you execute this command, you are prompted to provide:

- Valid administrator credentials
- RBS protocol (default: HTTP)
- RBS port (default: 2480)

Parameters

- **connect-options –** allows you to connect to a messaging server on startup. A monitor connection starts like a synchronization connection to the messaging server. Allowed values are:

Option	Description
`-u ml_username`	Required to connect to the messaging server.
`-p password`	Required to connect to the messaging server.

Option	Description

`-x [tcpip | tls | http | https] [(keyword=value;...)]`	Required to connect to the messaging server. The keyword=value pairs can be:
	• Host – the network name or IP address of the computer where the messaging server is running. By default, it is the computer where the monitor is running.
	• Protocol – should be set to the same network protocol and port as the messaging server is using for synchronization requests.
	• Additional Network Parameters – optional parameters, including:
	  • `buffer_size=number`
	  • `client_port=nnnn`
	  • `client_port=nnnn-mmmmm`
	  • `persistent=[0|1]` (HTTP and HTTPS only)
	  • `proxy_host=proxy_hostname` (HTTP and HTTPS only)
	  • `proxy_port=proxy_portnumber` (HTTP and HTTPS only)
	  • `url_suffix=suffix` (HTTP and HTTPS only)
	  • `version=HTTP-version-number` (HTTP and HTTPS only)

`-o outputfile.{mlm|csv}`	Closes the monitor at the end of the connection and saves the session in the specified file.

- **inputfile.{mlm|csv} –** prompts the monitor to open the specified file.

Package Administration Utilities

Deploy, delete, import, and export application packages using the administration command line utilities. You can execute these commands from the administration command line utility console or from the command line.

To issue commands using the interactive administration command line utility console, open `<<UnwiredPlatform_InstallDir>>\Servers\UnwiredServer\bin\supadmin.bat`. You must enter your host name, port, and administrator credentials before entering a command. Otherwise, the console prompts you for:

- Host – the host name for the Unwired Server. The default value is localhost.
- Port – the port used for the Unwired Server. The default value is 2000.
- Login ID – the administrator login ID to perform authentication with. The default value is supAdmin.

- Password – the administrator password to perform authentication with. The default value is s3pAdmin.

To issue the utility at the command line use:

```
supadmin.bat -host host name -port port -u username -pw password
commandName commandOptions
```

For example, to delete the package `test:1.0` from the domain, enter:

```
supadmin.bat -host test01.sybase.com -port 2000 -u supAdmin -pw
s3pAdmin delete -d default -pkg test:1.0
```

Deploy Application Package (deploy) Utility

Deploys a package to Unwired Server with the administration client APIs.

Syntax

```
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.
-dm *deploymentMode*	Sets the deployment mode. The mode determines how deployment handles the objects in a deployment unit and package. 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. • `NOCLOBBER` – deploys the package only if there are no objects in the target server's package that have the same name as any of those objects being deployed. • `REPLACE` – replaces any of the target objects with those in the package. After deployment, the servers package contains only those objects being deployed. • `VERIFY` – do not deploy package. Only return errors, if any. Used to determine the results of the `UPDATE` deploy mode.
-file *deploymentUnitFileName*	Defines the file name of the deployment unit. For example, `MyDeployUnit.xml`.

Option	Description
-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.
-sm *syncMode*	Defines the synchronization mode for the package; either `MBS` for messaging-based synchronization, or `REPLICATION` for replication-based synchronization.

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:

```
deploy   -file samples/uep/deployment/customer_list_unit.xml -dm
UPDATE -sl
```

Import Application Package (import) and Export Application Package (export) Utilities

The **import** command imports an existing package to Unwired Server. The **export** command exports a package from Unwired Server.

Syntax

```
import [import-options]
```

```
export [export-options]
```

Parameters

- **import-options | export-options –** use these options to import or export the package:

Option	Usage	Description
-d *domain*	import, export	During import, specifies the domain to which you are importing the package. During export, specifies the domain to which the package currently belongs.
-file *fileName*	import	Specifies the file name of the exported package.
-pkg *packageName*	export	Specifies the name of the package to export.

Examples

- **Import example –** This command imports the "samples/uep/deployment/customer_list_unit_copy.xml" file to the domain "default":

```
import   -d default -file samples/uep/deployment/
customer_list_unit_copy.xml
```

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

```
export   -d default -pkg samplePkg
```

Delete Application Package (delete) Utility

Deletes a package from Unwired Server.

Syntax

```
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:

```
delete   -d default -pkg samplePkg
```

Start and Stop sampledb Server (sampledb) Utility

Starts and stops the sampledb server from the command line.

Syntax

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

Parameters

- **install auto –** installs the sampledb server as a Windows service in auto-start mode.
- **install manual –** installs the sampledb server as a Windows service in manual start mode.
- **start –** starts the sampledb server.
- **stop –** stops the sampledb server.

Advantage Database Server Backup (adsbackup) Utility

A command line backup utility for use with Advantage Database Server. The **adsbackup** utility collects all the required information to perform a backup or restore, and executes the backup or restore on the specified server. This utility is located in *<UnwiredPlatform_InstallDir>*`\Servers\Advantage910\Server`.

Prerequisites: The **adsbackup.exe** utility loads `ACE32.DLL` or `libace.so` dynamically so it can be used with different versions of Advantage Client Engine (ACE). The minimum ACE version requirement is v6.0. However, if the server operating system is NetWare, the ACE version must be v7.0 or greater.

Syntax

To back up a directory of free tables:

```
adsbackup.exe [options] <src path> <file mask> <dest path>
```

To back up a data dictionary and its associated tables:

```
adsbackup.exe [options] <src database> <dest path>
```

Parameters

- **src path –** the path to the free tables you want to back up.
- **file mask –** a file mask similar to "*.adt" or "*.dbf" which determines which tables to include in the backup image.
- **dest path –** the backup image destination.
- **src database –** the path to the data dictionary you want to back up.
- **[options] –** these options are available:

Option	Description
-a	Creates a new backup image and initializes the source data so it can be used in a subsequent differential backup.
-c [ANSI|OMI]	Sets the character type of the SQL statement or connection used to open each table and transfer its data. Default: ANSI.
-d	Logs a warning before overwriting existing tables. The default behavior is to overwrite all tables when performing a backup or restore operation.
-f	Performs a differential backup. Includes only tables and records that have been modified since the last backup. Use this option only on databases on which you have previously called **ads-backup -a** to initialize the differential backup.
-h [ON|OFF]	Sets the rights-checking mode (used to open each table and transfer its data) of the SQL statement or connection used by the backup utility.
-i *<file1, file2...>*	Provides a list of tables to include in the backup or restore command. When using a data dictionary, specify the table name in the dictionary. When using free tables, specify the base table name including the file extension; for example, "table1.adt".
-e *<file1, file2...>*	Provides a list of tables to exclude from the backup or restore command. When using a data dictionary, specify the table name in the dictionary. When using free tables, specify the base table name including the file extension; for example, "table1.adt".
-m	Makes a backup or restore copy of only the data dictionary. Therefore, only metadata is backed up.
-n *<path>*	Specifies the location in which to store the backup log table.
-o *<filepath>*	Specifies a file name, along with the file path to the backup log table location.
-p	If the source path is a database, indicates the database password for the user. If the source path is a directory, lists the free table passwords. Free table usage can pass a single password for all encrypted tables, or a name=value pair for each table; for example, "password_for_all" or "table1=pass1;table2=pass2".
-q [PROP|COMPAT]	Specifies the locking mode of the SQL statement or connection used by the backup utility. This is the locking mode used to open each table and transfer its data; either proprietary (PROP) or compatible (COMPAT).
-r	Performs a restore instead of a backup (which is the default behavior).

Option	Description
-s *<server path>*	Specifies the connection path of the ADS server to perform the backup or restore operation. In most situations, **adsbackup** can determine the connection path to use by examining the source path. Use this option if you are having connection problems or want to use a specific connection path when performing the backup or restore.
-t *<server port>*	Defines the server port **adsbackup** connects to when using the Advantage JDBC Driver. This option is valid only with the Java **adsbackup** utility. Default: 6262.
-u [ADT|CDX|NTX]	Sets the table type of the SQL statement or connection used by the backup utility. This is the table type used to open each table and transfer its data. This also determines the table type of the log file produced by the backup or restore procedure.
-v [1-10]	Configures the lowest level of error severity to return. Default: 1.

Usage

You can use **adsbackup** in conjunction with the Unwired Server database file recovery tool **MORecover**. See *Sybase Unwired Platform Administration Guide > System Reference > Command Line Utilities > Unwired Server Runtime Utilities > Unwired Server Database File Recovery (MORecover) Utility*.

For more information on **adsbackup.exe** usage and sample use cases, see the *Advantage Database Server* documentation.

See also

- *Backing Up Messaging Data* on page 240

Unwired Server Database File Recovery (MORecover) Utility

The **MORecover** utility, with **adsbackup.exe**, backs up Unwired Server database files while the server is running. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\MessagingServer\Bin`.

Recent versions of the Advantage Database Server do not allow Unwired Server database files to be copied while the service is running. Stopping Unwired Server to perform regular backups is highly inconvenient in a production environment.

Prerequisite: Before executing **MORecover**:

1. Create an MORecover snapshot by running **adsbackup** to back up the tables required by MORecover. Use the **-i** (include) option to indicate the database tables to include in the backup. For an MORecover snapshot, these tables are:

- APPLICATIONS
- CFG_IDS
- CFG_PROP_VALUES
- CFG_SUBFOLDER_PROP_VALUES
- CFG_TEMPLATES
- DEVICES
- USER_DEVICE
- USERS

The source database argument for the command must be the full path of the `OBR.add` file in the Unwired Server data folder. The destination can be any folder. A collection of files constituting the backed-up data is created in the location you specify and stored as the MORecover backup data (see Example 1 below).

2. Convert the files generated by **adsbackup** into a standard database format. The generated files reflect the raw data from the original tables, excluding the index data. Therefore, files generated by **adsbackup** cannot be used directly as a database (see Example 2 below).

For information on **adsbackup** utility parameters and options, see *System Administration guide > System Reference > Command Line Utilities > Unwired Server Runtime Utilities > Advanatage Database Server Backup (adsbackup) Utility*.

Syntax

```
MORecover <recoveryDatabaseLocation>
```

Examples

- **Example 1: –** This example uses **adsbackup -i** (include) to indicate the database tables to include in the MORecover snapshot.

  ```
  adsbackup.exe -i
  APPLICATIONS,CFG_IDS,CFG_PROP_VALUES,CFG_SUBFOLDER_PROP_VALUES,CF
  G_TEMPLATES,DEVICES,USER_DEVICE,USERS <sourceDatabaseLocation>
  <destinationFileLocation>
  ```

- **Example 2: –** Before running MORecover to restore your database, you must convert the data files created by **adsbackup** back to a standard database format. The following example uses the **-r** option to restore the backup data to a temporary folder. You can then point MORecover to the restored data in this folder.

  ```
  adsbackup.exe -r <destinationFileLocation>
  <recoveryDatabaseLocation>
  ```

Usage

While the specified subset of tables typically results in a time-efficient backup operation, Sybase recommends that you schedule this command to run regularly during a time of light system load, since the backup operation contends with Unwired Server normal processing for database resources.

See also

- *Restoration of the Messaging Data* on page 242

Update Properties (updateprops.bat) Utility

Performs multiple functions, including registering or removing a participating node for a cluster or a relay server, or update a specific server property.

Syntax

```
updateprops.bat [-u username] [-p password] [-d dsn] [-f
propertyFile]
[-cn clusterName] [-nv "<propertyName=NewValue>"] [-r] [-v] [-x] [-
relayserver]
```

Parameters

- **-u *username* –** the platform administrator username.
- **-p *password* –** the platform administrator password.
- **-d *dsn* –** the data source name (DSN) of the cluster database.
- **-f *propertyFile* –** the name and path to the property that is going to be created or is used to change values. In the case of Unwired Platform this filed is likely to be `sup.properties`. This file is located in *`<UnwiredPlatform_InstallDir>`*`\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer`.
- **-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"
  ```
- **-r –** register a new Unwired Server (or relay server when used with the **-relayserver** option) to the cluster defined with -cn, or to copy cluster-affecting server changes from `sup.properties` to the cluster database. Administrators typically use this option if the cluster version differs from this Unwired Server. Ensure that the local files for this server are correct and there is no primary server that a slave server should be synchronized with.
- **-v –** use verbose output in the command window.
- **-x –** uninstall (remove) the component from the cluster.
- **-relayserver –** register or unregister the relay server or load balancer. must be used with either -r or -x, as required.

Examples

- **Removing a relay server from a cluster –** Remove the link between a relay server and an Unwired Server cluster by running this command on any participating server host:

updateProps.bat -x -relayserver

Note: This command does not remove the RSOE services. Only the relay server entries from cluster database and `sup.properties` are removed. To both unregister relay srver and remove RSOE services, run:

regRelayServer.bat remove

- **Removing an Unwired Server from a cluster –** Unregister an Unwired Server from the cluster by running this command:

  updateProps.bat -x
- **Adding an Unwired Server to a cluster –** Add an Unwired Server to an existing cluster by running this command after opening `sup.properties`, and changing the name of the cluster:

  updateProps.bat -r
- **Changing a cluster-affecting property –** Update the ml.threadcount property of the consolidated database to `20` by running:

  updateProps.bat -nv "ml.threadcount=20"

Usage

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

See also

- *Changing the Consolidated Database Server Thread Count and Pool Size* on page 79

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, like the relay server or cluster configuration, periodically check the configuration files for changes. For Unwired Server, administrators can instruct the server to reread the configuration files and apply the changes to the current process. However, some Unwired Server changes require that you restart the server.

Configure Unwired Server and its embedded subcomponents using Sybase Control Center, which then writes values to the appropriate file. For relay server, you must manually configure component files.

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 332
- *Relay Server Configuration Files* on page 336

Global Unwired Server Properties (sup.properties) Configuration File Reference

`sup.properties` is a global properties file that allows you to configure many subcomponents used by Unwired Server. This file is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer`.

Property	Default	Description
sup.admin.port	2000	The port used by the embedded application server.
sup.admin.protocol	IIOP	The protocol used for server administration (read-only).
sup.admin.httpports	8000	The HTTP ports used by the embedded application server.
sup.admin.httpsports	8001, 8002	The HTTPS (secure) ports used by the embedded application server.
sup.admin.iiopsport	2001	The IIOPS port used by the embedded application server.
sup.sync.sslkeystore	`Repository/Security/keystore.jks`	The relative path to the Unwired Server keystore file.
sup.sync.sslkeystore_password	changeit	The password to unlock the keystore file
sup.sync.ssltruststore	`Repository/security/truststore.jks`	The relative path to the Unwired Server truststore file.
sup.sync.ssltruststore_password	changeit	The password to unlock the truststore file

Property	Default	Description
sup.sync.port	2480	The synchronization port.
sup.sync.httpsport	2481	The secure synchronization port.
sup.sync.protocol	HTTP	The protocol used for synchronization. By default the protocol is HTTP; however, you can also use HTTPS.
sup.sync.certificate	n/a	The fully qualified path to the certificate file.
sup.sync.certificate_password	n/a	The password to unlock the certificate.
sup.cluster.name	n/a	The Unwired Platform cluster name.
sup.user.options	n/a	The Unwired Server user startup options.
sup.install.number	<number>	The Unwired Platform install number.
sup.java.install	n/a	The Unwired Platform Java install number.
sup.host	<computerName>	The host name of the SUP server.
sup.install.asservice	false	Indicates whether or not Unwired Server is installed in service mode.
sup.node.status	resumed	The Unwired Platform cluster status: suspend, resume, pending, suspended, resumed, pending.
sup.node.home	n/a	Mobile messaging service home path.
sup.imo.upa	n/a	Encrypted user name and password of admin@system.
sup.msg.outbound_queue_prefix	sup.mbs.moca.	The outbound queue prefix.
sup.msg.inbound_queue_prefix	sup.mbs.	The inbound queue prefix.
sup.msg.inbound_count	25	The number of inbound queues.
sup.msg.outbound_count	5	The number of outbound queues.
ml.threadcount	5	The synchronization threadcount. This value must be 5 units lower than sqlany.threadcount.
ml.servername	none	The Unwired Server name.
ml.cachesize	50M	The maximum size for the MobiLink server memory cache.

Property	Default	Description
relayserver.webserver.token	none	The Relay Server-backed Web server token. If left empty, a random value is generated.
relayserver.token	none	Replication-based synchronization backend server token. If empty, a random value is generated.
relayserver.https_port	443	The Relay Server HTTPS port.
relayserver.http_port	80	The Relay Server HTTP port.
relayserver.type	IIS	The Relay Server-hosted operating system type: IIS or Unix.
relayserver.farm_name	none	The Relay Server farm name. If empty, the value "*<ClusterName>*.SUPFarm" is used.
relayserver.msg.token	none	The Unwired Platform messaging backend server token. If empty, a random value is generated.
relayserver.trusted_certs	none	The trusted certificate path relative to Unwired Server home.
relayserver.protocol	HTTP	The Relay Server protocol: HTTP or HTTPS.
relayserver.host	Relayserver.sybase.com	The Relay Server host name.
relayserver.webserver.farm_name	none	The Relay Server Web server farm name. If empty, "*<ClusterName>*.SUPWebServerFarm" is used.
relayserver.msg.farm_name	none	The Relay Server messaging server farm name. If empty, "*<ClusterName>*.SUPMessagingFarm" is used.
cdb.threadcount	20	The maximum number of tasks that the consolidated database server can execute concurrently.
cdb.databasename	default	The consolidate database name.
cdb.password	sql	The password for the consolidated database.
cdb.asa.mode	primary	The database mode when consolidate database type is SQL Anywhere (Sybase_ASA).
cdb.serverport	5200	The consolidated database port number.
cdb.dsnname	default-cdb	The consolidated database DSN name.
cdb.install_type	default	The consolidated database install type.

Property	Default	Description
cdb.username	dba	The consolidated database user name.
cdb.type	Sybase_ASA	The consolidated database type.
cdb.serverhost	none	The consolidated database server host name.
cdb.user.options	none	The consolidated database user-specified startup options.
cdb.servername	none	The consolidated database server name.
cldb.serverhost	none	The cluster database server host name.
cldb.serverport	5200	The cluster database port number.
cldb.username	dba	The cluster database user name.
cldb.password	sql	The cluster database password.
cldb.databasename	clusterdb	The cluster database name.
cldb.dsnname	none	The cluster database DSN name.
cldb.type	Sybase_ASA	The cluster database type.
monitoringdb.server-host	none	The monitoring database server host name.
monitoringdb.server-port	5200	The monitoring database port number.
monitoringdb.user-name	dba	The monitoring database user name.
monitoringdb.pass-word	sql	The monitoring database password.
monitoringdb.databa-sename	monitordb	The monitoring database name.
monitoringdb.type	Sybase_ASA	The monitoring database type.
cluster.version	1	The cluster version number.
cluster.sync.share-dpath	none	The shared data path for CSYNC/DSYNC zip files; required only when "cluster.sync.share-dpath.enabled" is set to true.
cluster.sync.share-dpath.enabled	False	Indicates whether or not to enable the cluster's optional "Shared Data Path" feature.
license.product.edi-tion	none	The Unwired Platform license edition.

Property	Default	Description
license.type	none	The Unwired Platform license type.
client.licenses	none	The number of Unwired Platform device licenses.
client.url_suffix	none	The client URL suffix for replication-based synchronization when Relay Server is configured.
msg.http.server.ports	5001, 80	The Messaging Service HTTP ports.
msg.client.url_suffix	none	The client URL suffix for messaging-based synchronization when Relay Server is configured.
msg.admin.webservices.port	5100	The Messaging Service administration port.
msg.rsoeOptions	none	The Messaging Service RSOE options. Used in the Messaging Service RSOE command line.
msgserver.location	none	The file location of the Messaging Service.
webserver.rsoeOptions	none	The RSOE options for the embedded Web server requests (for example, data change notifications and DOE-C).
webserver.client.url_suffix	none	The client URL suffix for Web server requests when the Relay Server is configured (for example, data change notifications and DOE-C).
rsoeOptions	none	The RSOE options for the Messaging Service. Used in the replication-based synchronization RSOE command line.
sqlany.mode	Primary	The consolidated database mode: only primary is supported.
mac.address	none	The Mac address of the system.

Admin Security (default.xml) Configuration File Reference

Defines authentication and attribution properties for the 'admin' security configuration. The file is located in `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\CSI\conf`.

Note: Sybase recommends that you use Sybase Control Center to configure security settings, so that configuration changes are validated before being saved.

Define the login modules used for authentication requests. List the modules in the order that they are invoked with this syntax:

```
<config:authenticationProvider name="<authProviderName>"
controlFlag="<myValue>" />
```

See the *controlFlag* reference topic for possible configuration values. The default is `required`.

Configure global options if the same configuration is shared by the authentication and attribution providers by using:

```
<config:options name="<propertyName>" value="<myValue>" />
```

For an LDAP security provider, properties can include:

Property	Default	Description
ServerType	None	The LDAP server type. For example, `msad2k`, `sunone5`, `nsds4`, or `openldap`.
ProviderURL	`ldap://<host-name>:389`	The URL to connect to the LDAP server. For example, `ldap://localhost:389`. For SSL, use `ldap://localhost:636`.
SecurityProtocol	None	The protocol used to connect to the LDAP server. Sybase recommends that you set this to `SSL`. An SSL connection is required for Active-Directory when you set the password attribute.
InitialContextFactory	None	The factory class used to obtain initial directory context.
Referral	ignore	The behavior when a referral is encountered. The valid values are those dictated by LdapContext, for example, `ignore`, `follow`, or `throw`.
DefaultSearchBase	None	The default search base to use when performing general operations.

Property	Default	Description
AuthenticationSearchBase	None	The search base to use when performing authentication operations. If you do not set this value, the default search base is used.
SelfRegistrationSearchBase	None	The search base to use when creating a new user as part of self-registration. If you do not set this value, the authentication search base is used for self-update operations and the default search base is used for all other operations.
AuthenticationScope	ONELEVEL	Options include `ONELEVEL` or `SUBTREE`.
AuthenticationFilter	For most LDAP servers: `(&(uid={uid})(objectclass=person))` or For Active Directory e-mail lookups: `(&(userPrincipalName={uid})(objectclass=user)) [ActiveDirectory]` For Active Directory Windows user name lookups: `(&(sAMAccount-Name={uid})(objectclass=user))`	The user name and password authentication search filter. This must be a legal LDAP search filter, as defined in RFC 2254. The filter may contain the special string `"{uid}"` which is replaced with the user name of the user attempting to authenticate.

Property	Default	Description
CertificateAuthentication-Filter	For Active Directory server: `(&({certattr}={0})(object-class=user))` For most LDAP server types: `"(&({certattr}={0})(objectclass=person))"`	The certificate authentication search filter. The filter may contain the special string `"{certattr}"` which is replaced with the certificate attribute or the mapped LDAP attribute (if mapping between certificate attributes and LDAP attributes is defined). The value `"{0}"` is set to the encoded certificate or an attribute value from the certificate.
AuthenticationMethod	simple	The authentication method to use for all binding. Supported values are `DIGEST-MD5` or `simple`.
Attributes	None	Defines an attribute mapping from a CSI attribute to an LDAP attribute, including: • `CSI.Email` • `CSI.Username` • `CSI.Password`
BindDN	None	The DN to bind against when building the initial LDAP connection. The user being authenticated with the login module must have read permission on all user records.
BindPassword	None	The password to bind with for the initial LDAP connection. For example, `<config:options name="BindPassword" encrypted="true" value="1-AAAAEgQ-QyyYzC2+njB4K4QGPcMB1pM6XErTqZ1InyYrW/s56J69VfW5iBdFZeh-DrY66+6g9u1+a5VAqBiv/v5q08B3f59YMB1EQx9k93VgVTSC0w8q0="/>`.

Property	Default	Description
RoleSearchBase	None	The search base used to retrieve lists of roles. If this you do not set this value, the default search base is used.
RoleFilter	For SunONE/iPlanet: `(&(objectclass=ldapsubentry) (objectclass=nsroledefinition))` For Netscape Directory Server: `(objectclass=groupofnames) (objectclass=groupofuniquenames))` For ActiveDirectory: `(objectclass=groupofnames) (objectclass=group))`	When combined with the role search base and role scope, returns a complete list of roles within the LDAP server.
RoleScope	ONELEVEL	The role search scope. Options include `ONELEVEL` or `SUBTREE`.
RoleNameAttribute	cn	The attribute for retrieved roles that is the common name of the role.
UserFreeformRoleMembershipAttributes	None	The "freeform" role membership attribute list. Users who have attributes in this comma-delimited list are automatically granted access to roles that have names that are the same as the attribute value.

Property	Default	Description
UserRoleMembershipAttributes	The default value of this property depends on the chosen server type: • For SunONE 5.x, it is `"nsRoleDN"` • For ActiveDirectory, it is `"memberOf"`.	Defines the attributes that contain the DNs of all of the roles the user is a member of. These comma-delimited values are then cross-referenced with the roles retrieved from the role search base and search filter to create a list of user roles.
RoleMemberAttributes	There is a default value only for Netscape 4.x server: `"member,uniquemember"`	A comma-delimited list of potential attributes for roles. This list defines the DNs of people who are granted the specified roles. These values are cross-referenced with the active user to determine the user's roles.

Configure additional security providers using:

```
<config:provider name="<secProviderName>" type="<secType>" />
```

Security types include: `authorizer`, `attributer`, or `roleMapper`.

controlFlag Attribute Values

The Sybase implementation uses the same controlFlag values and definitions as those defined in the JAAS specification.

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

Control flag value	Description
(Default) required	The LoginModule is required to succeed. Authentication proceeds down the LoginModule list.
requisite	The LoginModule is required to succeed. 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 to succeed. Subsequent behavior depends on the authentication result: • If authentication succeeds, control returns immediately to the application (authentication does not proceed down the LoginModule list). • If authentication fails, authentication continues down the LoginModule list.
optional	The LoginModule is not required to succeed. Irrespective of success or failure, authentication proceeds down the LoginModule list.

Example

Say you list providers in the following order and set the corresponding controlFlag attributes as follows:

1. RADIUS token (required)
2. User name and password with native OS (sufficient)
3. User name and password via LDAP (optional)

This setup creates two tiers of authentication: the first tier is token-based, the second is login credential-based. A user must provide a valid RADIUS token to pass the first tier requirement. Next the user must enter a valid user name and password before passing both authentication challenges. There are two options: initially, the operating system attempts to validate the user name and password If that fails, LDAP attempts to validate the user.

In all cases, at least one LoginModule must succeed for authentication to be successful.

Unwired Server Logging (logging-configuration.xml) Configuration File

Defines the server logging configuration. This file is located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\Repository`

Note: Manual changes to the server log settings in one node of a cluster affect only the node on which the change is made. To modify cluster-wide server logging, you must individually edit the log file for each node.

Edit the `logging-configuration.xml` file to change server logging levels by component and define server file settings. The file syntax is:

```
<Entity EntityTypeId="<Logging Configuration>"
 <Entity EntityTypeId="LocalFileAppender">
  <EntityEntityTypeId="<UnwiredPlatformComponent>">
   <Prop name="LogLevel" value="<myLogLevel>" />
  </Entity>
  <Prop name="filename" value="<myFileName>"
  <Prop name="sizeRollover" value="<mySizeRollover>"
```

```
  <Prop name="maximumRolloverFiles" value="<myMaxRolloverFiles>"
  <Prop name="dateRollover" value="<myDateRollover>"
 </Entity>
</Entity>
```

The configurable properties are:

Property	Default	Description
LogLevel	The default log level varies depending upon the component: • Trace – TRACE • MMS – INFO • MSG – INFO • Security – INFO • MobiLink – INFO • DataServices - INFO • Other – WARN **Note:** If DOE-C is installed, it also appears as an Unwired Server component.	The log level for each configurable Unwired Platform component. Available log levels include: ALL, TRACE, DEBUG, INFO, WARN, ERROR, OFF, or CONSOLE. Trace is an interval component; do not modify it.
filename	logs/${sup.host}-server.log	The file name to which log data is written. Files are resolved relative to the server home directory.
sizeRollover	10MB	The number of bytes of granularity between rollover events. Supports optional KB, MB, and GB suffixes.
maximumRolloverFiles	1	The maximum number of rollover files that is maintained. Specify -1 for no limit.
dateRollover	NONE	The level of time granularity between rollover events: NONE, HOURLY, DAILY, WEEKLY, MONTHLY, YEARLY.

Runtime Message Tracing (TraceConfig.xml) Configuration File

Enables message tracing for various system components, and sets the tracing level. This file is located in *<UnwiredPlatform_InstallDir>*\Servers\MessagingServer\Data

The syntax is:

```
<TraceConfig>
            <Dir>\...UnwiredServer\logs</Dir>
            <Module Name="<moduleName>" Level="<traceLevel>"
```

```
Desc="<moduleDescription>" />
            ...
</TraceConfig>
```

The `<Dir>` property indicates the directory where tracing messages are stored when component tracing is enabled. The default location is `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\logs\<moduleName>` . Messages for each module are stored as .txt files in the corresponding folder.

The configurable properties are:

- Module name – the name of the system component.
- Level – the tracing level:

Level	Messages logged
All	Complete system information
Trace	Finer-grained informational events than debug
Debug	Very fine-grained system information, warnings, and all errors
Info	General system information, warnings, and all errors
Warn	Warnings and all errors
Error	Errors only
Off	None

- Desc – a unique identifying description for the module.

See also

- *Configuring Messaging and Mobile Workflow Runtime Logging* on page 230
- *Configuring Mobile Workflow Tracing* on page 231

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 320
- *Relay Server Configuration Files* on page 336

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

Configure properties for various Sybase Control Center services. These files are located in *<UnwiredPlatform_InstallDir>**SCC-XX*\services*<serviceName>* .

Key `service-config.xml` files include:

File	Description	Defaults
*<UnwiredPlatform_InstallDir>**SCC-XX*\services\RMI\service-config.xml	Sets the RMI agent port.	RMI port: 9999
*<UnwiredPlatform_InstallDir>**SCC-XX*\services\Messaging\service-config.xml	Sets the JMS messaging service port.	JMS messaging port: 2100
*<UnwiredPlatform_InstallDir>**SCC-XX*\services\SccSADataserver\service-config.xml	Sets the Sybase Control Center repository database port, and other properties.	Repository database port: 3683
*<UnwiredPlatform_InstallDir>**SCC-XX*\services\EmbeddedWebContainer\service-config.xml	Sets the Web container ports.	HTTP port: 8282 HTTPS port: 8283

Agent Plug-in Properties (agent-plugin.xml) Configuration File

Defines server properties for each Unwired Server to administer from Sybase Control Center. The file is located in *<UnwiredPlatform_InstallDir>**SCC-XX*\plugins\com.sybase.supadminplugin.

Properties include:

Property	Default	Description
auto.discovery.enable.on.startup	Personal Developer Edition: false Other editions: true	Enables or disables the automatic discovery of Unwired Servers when Sybase Control Center starts.

Property	Default	Description
auto.discovery.log.repeat	3	Repeats the logging of errors that are generated when a discovered Unwired Server is pinged. After the specified count is reached, Sybase Unified Agent stops printing error message to the log, but continues to ping the discovered server.
auto.discovery.schedule.enable.on.startup	true	Enables or disables scheduled Unwired Server discoveries. Scheduled discovery allows the agent to periodically check for newly installed Unwired Servers.
auto.discovery.schedule.interval	300000	Sets the scheduled discovery interval (in milliseconds).
sup.server.path	*<UnwiredPlatform_InstallDir>*\Servers\UnwiredServer	Sets the installation for Unwired Server. The path must be fully-qualified.
auto.register.localSUP.enable	true	If true, automatically registers the server as a managed resouce in Sybase Control Center. After launching and authenticating, registered resources automatically appear in the Unwired Platform console. If false, you need to manually register the server as managed resource in Sybase Control Center.

Sybase Control Center Logging (log4j.properties) Configuration File

Enables and configures Sybase Control Center logging. The log4j configuration file is located in *<UnwiredPlatform_InstallDir>**SCC-XX*\conf.

log4j properties include:

Property	Default	Description
log4j.appender.agent.File	<UnwiredPlatform_InstallDir>\SCC-XX\log\agent.log	The name and location of the Sybase Control Center log file.
log4j.appender.agent.MaxFileSize	5MB	The maximum size that a file can reach before a new one is created.
log4j.appender.agent.MaxBackupIndex	20	The number of log files that are backed up before the oldest file is deleted.

Role Mapping (roles-map.xml) Configuration File

The `roles-map.xml` file is located in *<UnwiredPlatform_InstallDir>\SCC-XX*\conf.

The `<uaf-roles>` section of the configuration file defines the available Sybase Control Center logical roles. The syntax is:

```
<uaf-roles>
   <role name="<myRoleName>" description="<myRoleDescription>"
</uaf-roles>
```

The `<security-modules>` section lists each login module defined in `csi.properties` file and maps the security provider's physical roles to the logical roles for Sybase Control Center. Specifically, the `SUP LDAP Login Module` default role mapping is:

```
<module name="SUP LDAP Login Module">
   <role-mapping modRole="SUP Administrator"
   uafRole="uaAnonymous,uaAgentAdmin,
   uaplug-inAdmin,sccAdminRole,sccUserRole" />
   <role-mapping modRole="SUP Domain Administrator"
   uafRole="uaAnonymous,uaAgentAdmin,uaplug-inAdmin,
   sccUserRole" />
 </module>
```

If necessary, replace the `modRole` values for each applicable login module with your own security provider role names to map them to Sybase Control Center roles. By default, the configuration file assumes that the security repository includes the "SUP Administrator" and "SUP Domain Administrator" roles. The "SUP Domain Administrator" role mapping is required only if you plan to assign domain administrators within the cluster.

Relay Server Configuration Files

Use relay server configuration files to set up the components of the relay server farm, and to define relay server host, port, protocol, and logging specifications.

You must manually configure the relay server using component files; you cannot access the server properties through the administration console.

See also

- *Unwired Server Configuration Files* on page 320
- *Sybase Control Center Configuration Files* on page 332
- *Relay Server Setup* on page 62
- *Relay Server Documentation* on page 63
- *Relay Server Utilities* on page 297
- *Configuring Relay Server Outbound Enabler Logging* on page 232

Relay Server (rs.config) Configuration File

Defines the relay server farm, including relay servers, back-end farms, and back-end servers.

When you register RSOE and relay server configuration changes by running **regRelayServer.bat**, the `rs.config` file is generated and stored in:

- For IIS: `C:\Inetpub\wwwroot\IAS_relay_server\Server\rs.config`
- For Apache: `<Apache_Home>/modules/rs.config`

This configuration file contains only the details of the server on which it is run, even if there is more than one server registered in the cluster. Therefore, you must ensure that all servers in the cluster are added to the file. The configuration file is organized into the following sections with the specified syntax:

- Relay server with auto-start option

  ```
  [options]
  start = no
  verbosity = 1
  ```

  When auto-start is enabled, the default log file is `%temp%\ias_relay_server.log` while the rshost is active. Upon server shutdown, the log file is renamed using the `YYMMDDNN.olg` file name format.

- Relay server peers

  ```
  [relay_server]
  enable = [yes|no]
  host = <myHostName>
  http_port = <myHttpPortNumber>
  https port = <myHttpsPortNumber>
  description = <myRSFarmMachineNumber>
  ```

- Backend farms

```
[backend_farm]
enable = [yes|no]
id = <myBackendFarmID>
client_security = [on|off]
backend_security = [on|off]
description = <myCompanyBackendFarm>
```

- Backend servers

```
[backend_server]
enable = [yes|no]
farm = <myFarmName>
id = <myBackendServerID>
mac = <myMACAddress>
token = <myBackendServerToken>
```

  You can uncomment the "mac" property to prompt the relay server to perform MAC verification. If you enter a value, it must be the MAC address that RSOE sends to the relay server. The MAC address is located in the `RSOE.log` file.

To apply the changes you make to this file, run this command from the directory where the `rs.config` file is located:

```
rshost -f rs.config -q -qc -u
```

See also

- *Configuring Relay Server in a Multinode Cluster* on page 66
- *Configuring a Relay Server for AfariaYou can configure a relay server for Afaria if you have deployed Afaria to a node separate from Unwired Server.*
- *Configuring and Enabling Relay Server Logging* on page 233

Relay Server Properties (relayserver.properties) Configuration File

Configures Unwired Platform client applications to synchronize through a relay server. The file is located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\config`.

Edit the file to configure the relay server. With Unwired Server running, register the relay server by running `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\bin\regRelayServer.bat`.

You can configure these properties for the relay server:

Property	Default	Description
relayserver.type	IIS	The relay server type: IIS or UNIX. The client connection string differs, depending on the server type.
relayserver.host	relayserver.sybase.com	The host name of the relay server load balancer.

Property	Default	Description
relayserver.http_port	80	The HTTP port of the relay server load balancer.
relayserver.https_port	443	The HTTPS port of the relay server load balancer.
relayserver.farm_name	None	The name of the Unwired Platform relay server farm, if applicable.
relayserver.token	None	(Optional) The security token used by the relay server to authenticate the back-end server connection. Specify a unique string to a maximum of 2048 characters.
relayserver.msg.farm_name	None	The messaging farm name.
relayserver.msg.token	None	(Optional) The security token used by the messaging server to authenticate the back-end server connection. Specify a unique string to a maximum of 2048 characters.
relayserver.webserver.farm_name	None	The Web server farm name. If you are configuring a farm using this property and the `relayserver.webserver.token` property, this farm is used for a cluster's DOE-C and DCN requests.
relayserver.webserver.token	None	(Optional) The security token used by the Web server to authenticate the back-end server connection. Specify a unique string to a maximum of 2048 characters.
relayserver.protocol	http	Indicates how clients synchronize data and how the relay server outbound enabler (RSOE) connects to the relay server; either `http` or `https`.

Property	Default	Description
relayserver.trusted_certs	None	Identifies the path to the trusted certificate file if the relay server uses a self-signed certificate or a certificate from an untrusted certificate authority (CA). If the server certificate is signed by a trusted CA, leave this property empty. **Note:** If you set this property, it must contain a file path relative to the installation directory for Unwired Server.
relayserver.rsoe.log.level	0	Configures the logging level for the RSOE. Valid values are: • 0 – log errors only. Use this logging level for deployment. • 1 – session-level logging. Provides an overview of a synchronization session. • 2 – request-level logging. Provides a more detailed view of HTTP requests within a synchronization session.

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

- *Monitoring Database* on page 14
- *Status and Performance Monitoring* on page 186
- *Setting Up an Existing Database for Monitoring* on page 78
- *Configuring Monitoring Performance Properties* on page 190

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.
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.

Column name	Column type	Description
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.

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.

Column name	Column type	Description
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.

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.

Column name	Column type	Description
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.
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.

Column name	Column type	Description
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.

mms_mbs_message Table

Detailed history information for message-based synchronization.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device performing the operation replay.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO).
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
receiveTime	timestamp	The received time of inbound message. Not applicable to outbound messages.
pushTime	timestamp	The pushed time of outbound message. Not applicable to inbound messages.
startTime	timestamp	The date and time the message was initiated.
endTime	timestamp	The date and time the message was completed.

Column name	Column type	Description
processTime	integer	The total time of the message, in milliseconds.
mbo	varchar(255)	The MBO performing the message.
operation	varchar(255)	The operation performing the message.
messageType	varchar(50)	The type of message. One of: SUBSCRIBE, UNSUBSCRIBE, OPERATION_REPLAY, RECOVER, SUSPEND, RESUME, RESUMENOREPLAY, IMPORT_DATA, DATA_RESET, LOGIN, UNKNOWN_TYPE.
isError	bit	Value is 1 if errors were recorded during transaction. 0 if no errors recorded.
payloadSize	integer	The size of the message payload.
isPushMsg	bit	Value is 1 if the message is outbound, 1 if otherwise.
isRequestMsg	bit	Value is 1 if the message is inbound, 0 if otherwise.
isSubscription	bit	Value is 1 if the message is a subscription request, 0 if not.
isOperationReplay	bit	Value is 1 if the message is a message-based operation replay, 0 if not.
sentPayloadSize	integer	The payload size of the outbound message.
receivedPayloadSize	integer	The payload size of the inbound message.
isMonitored	bit	Value is 1 if MBO is monitored, 0 if not.

Column name	Column type	Description
isLogged	bit	Value is 1 if the domain is logging data, 0 if not.

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.

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.

mms_data_change_notification Table

Information about data change notifications (DCNs) for messaging-based synchronization.

Column name	Column type	Description
packageName	varchar(255)	The package name of the mobile business object (MBO) affected by the DCN.
domain	varchar(255)	The domain to which the package involved in DCN belongs.
publicationName	varchar(255)	The synchronization group of the DCN.
notificationTime	timestamp	The time the DCN was sent.

Column name	Column type	Description
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.

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.
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_maxfullrefreshtime` `get_package_mbo_minfullrefreshtime`	• @packagename varchar(128) • mboname varchar (128) • startdate datetime • endate datetime	The minimum or maximum value of the duration for the mobile business object over the start date and end date range for a refresh activity.

Procedure	Parameters	Result
`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.

CHAPTER 11 Glossary: Sybase Unwired Platform

Defines terms for all Sybase Unwired Platform components.

administration perspective – Or administration console. The Unwired Platform administrative perspective is the Flash-based Web application for managing Unwired Server. *See* Sybase Control Center.

administrators – Unwired Platform users to which an administration role has been assigned. A user with the "SUP Administrator" role is called a "platform administrator" and a user with the "SUP Domain Administrator" role is called a "domain administrator". These administration roles must also be assigned SCC administration roles to avoid having to authenticate to Sybase Control Center in addition to Unwired Server:

- A domain administrator only requires the "sccUserRole" role.
- A platform administrator requires both the "sccAdminRole" and "sccUserRole" roles.

Adobe Flash Player – Adobe Flash Player is required to run Sybase Control Center. Because of this player, you are required to run Sybase Control Center in a 32-bit browser. Adobe does not support 64-bit browsers.

Advantage Database Server® – A relational database management system that provides the messaging database for Sybase Unwired Platform. *See* messaging database.

Afaria – An enterprise-grade, highly scalable device management solution with advanced capabilities to ensure that mobile data and devices are up-to-date, reliable, and secure. Afaria includes a server (Afaria Server), a database (Afaria Database), an administration tool (Afaria Administrator), and other runtime components.

Afaria stock channel – *See* stock channel.

APNS – Apple Push Notification Service.

artifacts – Artifacts can be client-side or automatically generated files; for example: `.xml`, `.cs`, `.java`, `.cab` files.

BAPI – Business Application Programming Interface. A BAPI is a set of interfaces to object-oriented programming methods that enable a programmer to integrate third-party software into the proprietary R/3 product from SAP. For specific business tasks such as uploading transactional data, BAPIs are implemented and stored in the R/3 system as remote function call (RFC) modules.

BLOB – Binary Large Object. A BLOB is a collection of binary data stored as a single entity in a database management system. A BLOB may be text, images, audio, or video.

cache – The virtual tables in the consolidated database that store synchronization data. *See* CDB.

cache group – Defined in Unwired WorkSpace, MBOs are grouped and the same cache refresh policy is applied to their virtual tables (cache) in the CDB.

cache partitions – Partitioning the cache divides it into segments that can be refreshed individually, which gives better system performance than refreshing the entire cache. Define cache partitions in Unwired WorkSpace by defining a partition key, which is a load parameter used by the operation to load data into the cache from the enterprise information system (EIS).

CDB – Consolidated database. The CDB stores runtime metadata (for Unwired Platform components) and cache data (for MBOs). *See also* data tier.

CLI – Command line interface. CLI is the standard term for a command line tool or utility.

client application – *See* mobile application.

client object API – The client object API is described in the *Developer Reference for BlackBerry*, *Developer Reference for iPhone*, and *Developer Reference for Windows Mobile*.

cluster – Also known as a server farm. Typically clusters are setup as either runtime server clusters or database clusters (also known as a data tier). Clustering is a method of setting up redundant Unwired Platform components on your network in order to design a highly scalable and available system architecture.

cluster database – A data tier component that holds information pertaining to all Unwired Platform server nodes. Other databases in the Unwired Platform data tier includes the consolidated, messaging, and monitoring databases, and the database for Afaria.

connection – Includes the configuration details and credentials required to connect to a database, Web service, or other EIS.

connection pool – A connection pool is a cache of Enterprise Information System (EIS) connections maintained by Unwired Server, so that the connections can be reused when Unwired Server receives future requests for data.

connection profile – In Unwired WorkSpace, a connection profile includes the configuration details and credentials required to connect to an EIS.

context variable – In Unwired WorkSpace, these variables are automatically created when a developer adds reference(s) to an MBO in a mobile application. One table context variable is created for each MBO attribute. These variables allow mobile application developers to specify form fields or operation parameters to use the dynamic value of a selected record of an MBO during runtime.

data change notification (DCN) – Data change notification (DCN) allows an Enterprise Information System (EIS) to synchronize its data with the consolidated database through a push event.

data refresh – A data refresh synchronizes data between the consolidated database and a back-end EIS so that data in the cache is updated. *See also* scheduled data refresh.

data source – In Unwired WorkSpace, a data source is the persistent-storage location for the data that a mobile business object can access.

data tier – The data tier includes Unwired Server data such as cache, cluster information, and monitoring. The data tier includes the consolidated database (CDB), cluster, monitoring, and messaging databases.

deploy – (Unwired Server) Uploading a deployment archive or deployment unit to an Unwired Server instance. Unwired Server can then make these units accessible to users via a client application that is installed on a mobile device.

There is a one-to-one mapping between an Unwired WorkSpace project and a server package. Therefore, all MBOs that you deploy from one project to the same server are deployed to the same server package.

deployment archive – In Unwired WorkSpace, a deployment archive is created when a developer creates a package profile and executes the **build** operation. Building creates an archive that contains both a deployment unit and a corresponding descriptor file. A deployment archive can be delivered to an administrator for deployment to a production version of Unwired Server.

deployment descriptor – A deployment descriptor is an XML file that describes how a deployment unit should be deployed to Unwired Server. A deployment descriptor contains role-mapping and domain-connection information. You can deliver a deployment descriptor and a deployment unit—jointly called a deployment archive—to an administrator for deployment to a production version of Unwired Server.

deployment mode – You can set the mode in which a mobile application project or mobile deployment package is deployed to the target Unwired Server.

deployment profile – A deployment profile is a named instance of predefined server connections and role mappings that allows developers to automate deployment of multiple packages from Sybase Unwired WorkSpace to Unwired Server. Role mappings and connection mappings are transferred from the deployment profile to the deployment unit and the deployment descriptor.

deployment unit – The Unwired WorkSpace build process generates a deployment unit. It enables a mobile application to be effectively installed and used in either a preproduction or production environment. Once generated, a deployment unit allows anyone to deploy all required objects, logical roles, personalization keys, and server connection information together, without requiring access to the whole development project. You can deliver a deployment unit and a deployment descriptor—jointly called a deployment archive—to an administrator for deployment to a production version of Unwired Server.

development package – A collection of MBOs that you create in Unwired WorkSpace. You can deploy the contents of a development package on an instance of Unwired Server.

device application – *See also* mobile application. A device application is a software application that runs on a mobile device.

Device Application Designer – In Unwired WorkSpace, this is the Eclipse plug-in that you can use to create and edit custom device applications for device clients graphically.

device notification – Replication-based synchronization (RBS) clients receive device notifications when a data change is detected for any of the MBOs in the synchronization group to which they are subscribed. Both the change detection interval of the synchronization group and the notification threshold of the subscription determine how often RBS clients receive device notifications. Administrators can use subscription templates to specify the notification threshold for a particular synchronization group.

device user – The user identity tied to a device.

DML – Data manipulation language. DML is a group of computer languages used to retrieve, insert, delete, and update data in a database.

DMZ – Demilitarized zone; also known as a perimeter network. The DMZ adds a layer of security to the local area network (LAN), where computers run behind a firewall. Hosts running in the DMZ cannot send requests directly to hosts running in the LAN.

domain administrator – A user to which the platform administrator assigns domain administration privileges for one or more domain partitions. The domain administrator has a restricted view in Sybase Control Center, and only features and domains they can manage are visible.

domains – Domains provide a logical partitioning of a hosting organization's environment, so that the organization achieves increased flexibility and granularity of control in multitenant environments. By default, the Unwired Platform installer creates a single domain named "default". However the platform administrator can also add more domains as required.

EIS – Enterprise Information System. EIS is a back-end system, such as a database.

Enterprise Explorer – In Unwired WorkSpace and Device Application Designer, Enterprise Explorer allows you to define data source and view their metadata (schema objects in case of database, BAPIs for SAP, and so on).

export – The Unwired Platform administrator can export the mobile objects, then import them to another server on the network. That server should meet the requriement needed by the exported MBO.

hostability – *See* multitenancy.

IDE – Integrated Development Environment.

JDE – BlackBerry Java Development Environment.

key performance indicator (KPI) – Used by Unwired Platform monitoring. KPIs are monitoring metrics that are made up for an object, using counters, activities, and time which

jointly for the parameters that show the health of the system. KPIs can use current data or historical data.

keystore – The location in which encryption keys, digital certificates, and other credentials in either encrypted or unencrypted keystore file types are stored for Unwired Server runtime components. *See also* truststore.

LDAP – Lightweight Directory Access Protocol.

local business object – Defined in Unwired WorkSpace, local business objects are not bound to EIS data sources, so cannot be synchronized. Instead, they are objects that are used as local data store on device.

logical role – Logical roles are defined in mobile business objects, and mapped to physical roles when the deployment unit that contain the mobile business objects are deployed to Unwired Server.

matching rules – A rule that triggers a mobile workflow application. Matching rules are used by the mobile workflow email listener to identify e-mails that match the rules specified by the administrator. When emails match the rule, Unwired Server sends the e-mail as a mobile workflow to the device that matches the rule. A matching rule is configured by the administrator in Sybase Control Center.

MBO – Mobile business object. The fundamental unit of data exchange in Sybase Unwired Platform. An MBO roughly corresponds to a data set from a back-end data source. The data can come from a database query, a Web service operation (including Remedy), or SAP. An MBO contains both concrete implementation-level details and abstract interface-level details. At the implementation-level, an MBO contains read-only result fields that contain metadata about the data in the implementation, and parameters that are passed to the back-end data source. At the interface-level, an MBO contains attributes that map to result fields, which correspond to client properties. An MBO may have operations, which can also contain parameters that map to arguments, and which determines how the client passes information to the enterprise information system (EIS).

You can define relationships between MBOs, and link attributes and parameters in one MBO to attributes and parameters in another MBO.

MBO attribute – An MBO attribute is a field that can hold data. You can map an MBO attribute to a result field in a back-end data source; for example, a result field in a database table.

MBO binding – An MBO binding links MBO attributes and operations to a physical data source through a connection profile.

MBO operation – An MBO operation can be invoked from a client application to perform a task; for example, create, delete, or update data in the EIS.

MBO relationship – MBO relationships are analogous to links created by foreign keys in a relational database. For example, the account MBO has a field called *owner_ID* that maps to the *ID* field in the owner MBO.

Define MBO relationships to facilitate:

- Data synchronization
- EIS data-refresh policy

messaging based synchronization (MBS) – A synchronization method where data is delivered asynchronously using a secure, reliable messaging protocol. MBS provides fine-grained synchronization (synchronization is provided at the data level—each process communicates only with the process it depends on), and it is therefore assumed that the device is always connected and available. *See also* replication based synchronization.

messaging database – The messaging database allows in-flight messages to be stored until they can be delivered. This database is used in a messaging based synchronization environment. The messaging database is part of the Unwired Platform data tier, along with the consolidated, cluster, and monitoring databases.

mobile application – A Sybase Unwired Platform mobile application is an end-to-end application, which includes the MBO definition (back-end data connection, attributes, operations, and relationships), the generated server-side code, and the client-side application code.

Mobile Application Diagram – The Mobile Application Diagram is the graphical interface to create and edit MBOs. By dragging and dropping a data source onto the Mobile Application Diagram, you can create a mobile business object and generate its attribute mappings automatically.

Mobile Application Project – A collection of MBOs and client-side, design-time artifacts that make up a mobile application.

Mobile client – *See* mobile application client.

mobile workflow packages – Mobile workflow packages use the message-based synchronization model. The mobile workflow packages are deployed to Unwired Server, and can be deployed to mobile devices, via the Unwired Platform administrative perspective in Sybase Control Center.

monitoring – Monitoring is an Unwired Platform feature available in Sybase Control Center that allows administrators to identify key areas of weakness or periods of high activity in the particular area they are monitoring. It can be used for system diagnostic or for troubleshooting. Monitored operations include replication-based synchronization, messaging-based synchronization, messaging queue, data change notification, device notification, package, user, and cache activity.

monitoring database – A database that exclusively stores data related to replication and messaging synchronization, queues status, users, data change notifications, and device

notifications activities. By default, the monitoring database runs in the same data tier as the consolidated database, messaging database and cluster database.

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

multitenancy – The ability to host multiple tenants in one Unwired Cluster. Also known as hostability. *See also* domains.

node – A host or server computer upon which one or more runtime components have been installed.

object query – Defined in Unwired WorkSpace for an MBO and used to filter data that is downloaded to the device.

openDS – The default LDAP server that is installed in Developer Edition and is suitable for authentication and authorization in a development environment.

operation – *See* MBO operation.

package – A package is a named container for one or more MBOs. On Unwired Server a package contains MBOs that have been deployed to this instance of the server.

palette – In Unwired WorkSpace, the palette is the graphical interface view from which you can add MBOs, local business objects, structures, relationships, attributes, and operations to the Mobile Application Diagram.

parameter – A parameter is a value that is passed to an operation/method. The operation uses the value to determine the output. When you create an MBO, you can map MBO parameters to data-source arguments. For example, if a data source looks up population based on a state abbreviation, the MBO gets the state from the user, then passes it (as a parameter) to the data source to retrieve the information. Parameters can be:

- Synchronization parameters – synchronize a device application based on the value of the parameter.
- Load parameters – perform a data refresh based on the value of the parameter.
- Operation parameters – MBO operations contain parameters that map to data source arguments. Operation parameters determine how the client passes information to the enterprise information system (EIS).

personalization key – A personalization key allows a mobile device user to specify attribute values that are used as parameters for selecting data from a data source. Personalization keys are also used as operation parameters. Personalization keys are set at the package level. There are three type of personalization keys: Session, client, server.

They are most useful when they are used in multiple places within a mobile application, or in multiple mobile applications on the same server. Personalization keys may include attributes such as name, address, zip code, currency, location, customer list, and so forth.

physical role – A security provider group or role that is used to control access to Unwired Server resources.

Problems view – In Eclipse, the Problems view displays errors or warnings for the Mobile Application Project.

provisioning – The process of setting up a mobile device with required runtimes and device applications. Depending on the synchronization model used and depending on whether or not the device is also an Afaria client, the files and data required to provision the device varies.

pull synchronization – Pull synchronization is initiated by a remote client to synchronize the local database with the CDB. On Windows Mobile, pull synchronization is supported only in RBS applications.

push synchronization – Push is the server-initiated process of downloading data from Unwired Server to a remote client, at defined intervals, or based upon the occurrence of an event.

queue – In-flight messages for a messaging application are saved in a queue. A queue is a list of pending activities. The server then sends messages to specific destinations in the order that they appear in the queue. The depth of the queue indicates how many messages are waiting to be delivered.

relationship – *See* MBO relationship.

relay server – *See also* Sybase Hosted Relay Service. The relay server, a required component in a highly-available production environment, adds an extra layer of security and load balancing to the server environment. This server component, which is deployed into the enterprise DMZ, allows secure communications between devices and Unwired Server components across the firewall.

replication based synchronization (RBS) – A synchronization method where data is delivered synchronously using an upload/download pattern. For push-enabled clients, RBS uses a "poke-pull" synchronization model, where a notification is pushed to the device (poke), and the device fetches the content (pull), and is assumed that the device is not always connected to the network and can operate in a disconnected mode and still be productive. For clients that are not push-enabled, the default synchronization model is pull.*See also* messaging based synchronization.

REST web services – Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

RFC – Remote Function Call. You can use the RFC interface to write applications that communicate with SAP R/3 applications and databases. An RFC is a standalone function. Developers use SAP tools to write the Advanced Business Application Programming (ABAP) code that implements the logic of a function, and then mark it as "remotely callable," which turns an ABAP function into an RFC.

role – *See also* logical role and physical role. Roles control access to Sybase Unwired Platform resources.

role mapping – Maps a physical (server role) to a logical (Unwired Platform role). Role mappings can be defined by developers, when they deploy an MBO package to a development Unwired Server, or by platform or domain administrators when they assign a security configuration to a domain or deploy a package to a production Unwired Server (and thereby override the domain-wide settings in the security configuration).

RSOE – Relay Server Outbound Enabler. An RSOE is an application that manages communication between a back-end server—Unwired Server or Afaria—and a relay server.

runtime server – An instance of Unwired Server that is running. Typically, a reference to the runtime server implies a connection to it.

SAP – SAP is one of the EIS types that Unwired Platform supports.

SCC – Sybase Control Center. A Web-based interface that allows you to administer your installed Sybase products.

scheduled data refresh – Data is updated in the consolidated database from a back-end EIS, based on a scheduled data refresh. Typically, data is retrieved from an EIS (for example, SAP) when a device user synchronizes. However, if an administrator wants the data to be preloaded for a mobile business object, a data refresh can be scheduled so that data is saved locally in a cache. By preloading data with a scheduled refresh, the data is available in the information server when a user synchronizes data from a device. Scheduled data refresh requires that an administrator define a cache group as "scheduled" (as opposed to "on-demand").

security configuration – Part of the application user and administration user security. A security configuration determines the scope of user identity, authentication and authorization checks, and can be assigned to one or more domains by the platform administrator in Sybase Control Center. A security configuration contains:

- A set of configured security providers (for example LDAP) to which authentication, authorization, attribution is delegated.
- Role mappings (which can be specified at the domain or package level)

security provider – A security provider and it's repository holds information about the users, security roles, security policies, and credentials used by some to provide security services to Unwired Platform. A security provider is part of a security configuration.

security profile – Part of the Unwired Server runtime component security. A security profile includes encryption metadata to capture certificate alias and the type of authentication used by server components. By using a security profile, the administrator creates a secured port over which components communicate.

server connection – The connection between Unwired WorkSpace and a back-end EIS is called a server connection.

server farm – *See also* cluster. Is the relay server designation for a cluster.

server-initiated synchronization – *See* push synchronization.

SOAP – Simple Object Access Protocol. SOAP is an XML-based protocol that enables applications to exchange information over HTTP. SOAP is used when Unwired Server communicates with a Web service.

solution – In Visual Studio, a solution is the high-level local workspace that contains the projects users create.

Solution Explorer – In Visual Studio, the Solution Explorer pane displays the active projects in a tree view.

statistics – In Unwired Platform, the information collected by the monitoring database to determine if your system is running as efficiently as possible. Statistics can be current or historical. Current or historical data can be used to determine system availability or performance. Performance statistics are known as key performance indicators (KPI).

Start Page – In Visual Studio, the Start Page is the first page that displays when you launch the application.

structured data – Structured data can be displayed in a table with columns and labels.

structure object – Defined in Unwired WorkSpace, structures hold complex datatypes, for example, a table input to a SAP operation.

subscription – A subscription defines how data is transferred between a user's mobile device and Unwired Server. Subscriptions are used to notify a device user of data changes, then these updates are pushed to the user's mobile device.

Sybase Control Center – Sybase Control Center is the Flash-based Web application that includes a management framework for multiple Sybase server products, including Unwired Platform. Using the Unwired Platform administration perspective in Sybase Control Center, you can register clusters to manage Unwired Server, manage domains security configurations, users, devices, connections and monitor the environment. You can also deploy MBO packages and manage deployed MBO packages in order to design the synchronization behavior for those packages. Only use the features and documentation for Unwired Platform. Default features and documentation in Sybase Control Center do not always apply to the Unwired Platform use case.

Sybase Hosted Relay Service – The Sybase Hosted Relay Service is a Web-hosted relay server that enables you to test your Unwired Platform development system.

Sybase Messaging Service – The synchronization service that facilitates communication with device client applications.

Sybase Unified Agent – Provides runtime services to manage, monitor, and control distributed Sybase resources. The agent must be running for Sybase Control Center to run.

Sybase Unwired Platform – Sybase Unwired Platform is a development and administrative platform that enables you to mobilize your enterprise. With Unwired Platform, you can

develop mobile business objects in the Unwired WorkSpace development environment, connect to structured and unstructured data sources, develop mobile applications, deploy mobile business objects and applications to Unwired Server, which manages messaging and data services between your data sources and your mobile devices.

Sybase Unwired WorkSpace – Sybase Unwired Platform includes Unwired WorkSpace, which is a development tool for creating mobile business objects and mobile applications.

synchronization group – Defined in Unwired WorkSpace, a synchronization group is a collection of MBOs that are synchronized at the same time.

synchronization parameter – A synchronization parameter is an MBO attribute used to filter and synchronize data between a mobile device and Unwired Server.

synchronization phase – For replication based synchronization packages, the phase can be an upload event (from device to the consolidated database) or download event (from the consolidated database to the device).

synchronize – *See also* data refresh. Synchronization is the process by which data consistency and population is achieved between remote disconnected clients and Unwired Server.

truststore – The location in which certificate authority (CA) signing certificates are stored. *See also* keystore.

undeploy – Running **undeploy** removes a domain package from an Unwired Server.

Unwired Server – The application server included with the Sybase Unwired Platform product that manages mobile applications, back-end EIS synchronization, communication, security, transactions, and scheduling.

user – Sybase Control Center displays the mobile-device users who are registered with the server.

Visual SQL – A graphical user interface tool that you can use to build SQL queries.

Visual Studio – Microsoft Visual Studio is an integrated development environment product that you can use to develop device applications from generated Unwired WorkSpace code.

Welcome page – In Eclipse, the first set of pages that display when you launch the application.

workspace – In Eclipse, a workspace is the directory on your local machine where Eclipse stores the projects that you create.

WorkSpace Navigator – In Eclipse, the tree view that displays your mobile application projects.

WSDL file – Web Service Definition Language file. The file that describes the Web service interface that allows clients to communicate with the Web service. When you create a Web

service connection for a mobile business object, you enter the location of a WSDL file in the URL.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W