Security

Sybase Unwired Platform 2.1 ESD #3

Contents

CHAPTER 1 Documentation Roadmap for Unwired Platform

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

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

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

CHAPTER 2 Introduction to Security

Mobility has changed the computing and network environments of today. Before mobility, enterprise security primarily focused on the firewall and limited access to digital assets to only those users authenticated within the enterprise information system (EIS).

An Unwired Platform deployment introduces a multilayer approach to corporate security designed for mobility. This approach ensures that:

- Internal and external device users can securely connect to enterprise information systems.
- Every network link that transfers corporate information and every location that stores enterprise data guarantees confidentiality.

Before you can prepare for the scale and scope of activities required to secure Unwired Platform, learn which components you can secure, which communication streams you can protect, and how you can control access to mobile digital assets:

Component Security

Unwired Platform consists of multiple components that are installed on internal networks, primarily on the corporate LAN and the demilitarized zone (DMZ). Each component requires specific administration tasks to secure it.

Review this diagram to understand where platform components are installed, then review the

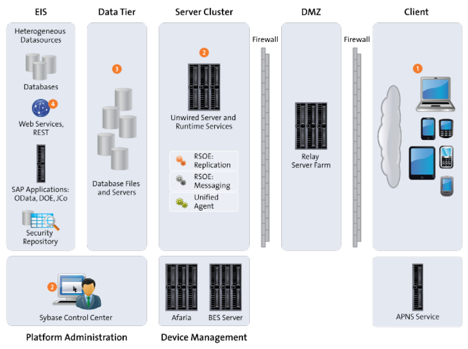


table to understand how they are secured.

Numbers in this diagram identify various Sybase Unwired Platform security features. Some features are standards of the platform, while others are optional and up to the administrator or developer to implement.

Component	How Secured
1. Mobile application and local data	• Login screens • Encryption of local data • Initial provisioning • Remote administration and security features See Device Security.
2. Unwired Server and runtime data services	• Authentication of users and administrators • Enforcing device registration • Secure communication to subcomponents • Secure administration of server and services See Server Security.
3. Cache (CDB) and messaging database	• Encryption of data and logs • Supplying custom password during install; changing changing of installer-defined passwords supported See Data Tier Security.
4. Enterprise Information Servers (EIS)	• Secure connections • Secure data change notifications See EIS Security.

Communication Security

Secure Unwired Platform component communications to prevent packet sniffing or data tampering. Different combinations of components communicate with different protocols and different ports.

Note: As an alternative to reading all the topics on communication security, use *Port Number Reference* as a quick reference on all ports and information on how to change them.

See also

- *Port Number Reference* on page 187

Device-to-Platform Communications

Depending on your environment, devices typically connect to a Relay Server deployed to the DMZ (recommended). Alternatively, you can use a third reverse proxy or load balancers. However, in most Unwired Platform deployments, a Relay Server is the first line of defense to the platform by acting as a proxy for the device, and facilitating interactions with Unwired Servers installed on the corporate LAN.

- For Relay Server connections, the traffic content depends on the payload protocol of the application:
  - Messaging communication encrypts the entire communication stream with a proprietary protocol.
  - Replication communication uses both HTTP or HTTPS protocol

  For Relay Server, configure the Web server host (IIS or Apache) to use a secure port, and use Sybase Control Center to configure Relay Server to use secure protocols, ports, and certificates in Sybase Control Center.

  On the client (in the case of mutual authentication), install certificates, and configure profiles to connect to Relay Server.
- Reverse proxies or load balancers require you to open firewall holes from the DMZ to Unwired Platform, but the same protocols described for messaging and replication still apply.

See also

- *Port Number Reference* on page 187

Unwired Server and Device Application Communications

Unwired Server communicates differently with replication, messaging, or Gateway applications.

- Replication applications – can use HTTP or HTTPS. By default, the data content in HTTP is unencrypted but compressed. HTTPS keeps the data confidential. For additional security, application developers can add end-to-end encryption (E2EE), in which all the data is encrypted between the device and Unwired Server. By default, the installer generates a keypair that is used by all installations of Unwired Platform. Therefore you must replace these defaults to avoid compromising security. You can generate new ones, then replace these keypairs in Sybase Control Center. You must then provision the public client key to the device, and configure the device connection profile with the key location. Only then is data encrypted using an AES in cipher-block chaining mode; RSA handles the key exchange. See *Encrypting Synchronization for Replication Payloads*.
- Messaging applications, hybrid workflow apps, and Online Data Proxy/OData applications – network traffic uses HTTP or HTTPS except in the case of Windows Mobile Hybrid Web Container (which does not support HTTPS). Each HTTP message contains an encrypted message and follows this process:
  1. When the messaging server is installed, it generates an RSA key pair.
  2. When a device first contacts the server, the device retrieves the public key and the server uses it to secure all future communication. For performance reasons, only a small section of the data from device to server is encrypted with the public key. Other items of note:
     - Administrators can enable autoregistration by setting up an application connection template in Sybase Control Center. Automatic registration means that administrators need neither set up whitelists nor generate single-use passwords.

For details, see *Automatically Registering Applications* in Sybase Control Center online help.

- For ODATA applications, developers can use Afaria to preprovision the RSA public key to the client application.

  However, in non-Afaria environments or for non-ODATA applications excluding those for BlackBerry, Sybase requires that you initially install the messaging application, and connect directly to the messaging service on the corporate LAN (via WiFi, cradle, and so on). Once initial registration is complete, the device can be used outside of the LAN by substituting the connection profile properties to use the internet accessible (typically, Relay Server) addresses. See *Provisioning Security Artifacts*

  For BlackBerry devices, because the BES is already inside the LAN, the initial provisioning of the RSA public key is considered safe.

3. Registration adds the user name and authorization code to a whitelist. When the messaging client connects to the messaging server, it passes the user name, the activation code, and the device and application IDs to the server. The device ID is derived from the hardware. The user name, application ID, and device ID uniquely identify the registration.
4. The device identified by the DeviceID is permanently assigned to that user and added to the white list. For every future interaction, a communication session is initialized using the public key. For the remainder of the session, a rotating sequence of AES keys is used to yield better performance.

All data transferred between the device and the Messaging Server is encrypted in this manner.

See also

- *Provisioning Security Artifacts* on page 132
- *Encrypting Synchronization for Replication Payloads* on page 102
- *Port Number Reference* on page 187

Unwired Server and Device Push Notifications

Sometimes Unwired Server must asynchronously notify a device application of changes it should be aware of. The mechanism for transmitting a push notification depends on the device type.

The notification protocol depends on platform:

- For iOS devices, Unwired Server uses the APNS service. See *BlackBerry Provisioning with BES* in *System Administration*.
- For BlackBerry devices, use the HTTP gateway push features of the MDS servers to deliver the notification. See *BlackBerry Push Notification Properties* in Sybase Control Center online help.

- For other types, use target change notifications (push notifications) in Unwired Server. See *Setting up Push Synchronization* in *System Administration*.

Unwired Server and Data Tier Communications

Unwired Server uses two synchronization databases for its data tier: one for replication and one for messaging. It also connects to a cluster, monitor, and domain log database.

All communication streams between Unwired Platform and databases comprising are unencrypted, because they are exchanged on the corporate LAN. Nontheless, ensure that the local subnet is protected from network sniffing and file access.

See also

- *Port Number Reference* on page 187

Unwired Server and Sybase Control Center Communications

There are two different communication streams used to communicate with the Sybase Control Center administration tool: one for communications with the Sybase Control Center web console, and one for the communications with the Sybase Control Center X.X Windows service.

Communications between Unwired Server and the Sybase Control Center X.X Windows service use IIOPS on port 2001 by default. While Unwired Platform installs a sample certificate to enable the use of IIOPS automatically, you should exchange the certificate with a production-ready one immediately following installation.

There are two self-signed certificates that need to be changed: one for Unwired Server, and one for Sybase Control Center.

See also

- *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners* on page 41
- *Port Number Reference* on page 187

Unwired Server and EIS Communications

Secure communication between Unwired Server and any supported back-end depends on the direction of the interaction between these components.

- EIS to Unwired Server – can communicate only via the DCN feature. DCN uses HTTP or HTTPS and all requests are also authenticated via the DCN User role. Sybase discourages the use of HTTP for DCN: Unwired Server challenges the EIS for BASIC authentication credentials; these could be intercepted by network sniffers in HTTP. Consequently, Sybase strongly recommends the use of HTTPS. Further, if you are creating applications that connect over Online Data Proxy or DOE, you can enable mutual authentication between these components as required. To secure the channel:

    - The EIS developer uses HTTPS to construct and send DCN requests to the listener.
    - The administrator manages the certificates, then uses Sybase Control Center to configure an HTTPS listener for DCNs. If you are connecting with Online Data Proxy or a DOE-C, then each type of connection requires it's own security profile, and the DCN listener profile should not be used in this case.
- Unwired Server to EIS – Unwired Server can perform operation replays. The manner in which those replays are communicated depends on the EIS and whether or not the administrator secures this channel in Sybase Control Center:
    - REST/SOAP uses BASIC authorized over HTTP or HTTPS.
    - REST/SOAP for SAP® uses BASIC/SSO2/X.509 authentication over HTTP or HTTPS.
    - JCo for SAP uses one of username/password, SSO2 tokens, or X.509 over SNC.
    - JDBC uses driver specific mechanisms to encrypt traffic. Review your JDBC driver documentation to learn how to configure this.

  For each of these EIS communication channels, you must configure the secure protocol. Otherwise, the user's credentials can potentially be exposed network sniffers. Once you have configured the secure channel, always ensure that EIS server certificates are imported into the Unwired Server truststore to allow this communication.

See also

- *Securing Data Change Notifications* on page 139
- *Port Number Reference* on page 187

Unwired Server Nodes in Production Cluster Communications

Unwired Servers communicate with other Unwired Servers in the same cluster differently, depending on the type of communication performed.

- For replication synchronization, the servers use the secure replication ports to negotiate which server acts as the primary synchronization server.
- For cluster and domain configuration synchronization, the servers use HTTP listeners. If a shared disk is used, the servers do not use any network protocol for the file transfers; instead, they just access the disk.
- For the exchange JMS messages, servers use IIOPS.
- For other shared data or information, communicate indirectly using a shared databases on the data tier.

See also

- *Port Number Reference* on page 187

Authentication and Access Security

Authentication and role-based access control (RBAC) are core security features supported by all application types to control access to enterprise digital assets. Review key concepts of authentication and role-based access control in Unwired Platform.

Security Provider Plug-in Model

Implement authentication and access control with the Common Security Infrastructure (CSI) component. Use CSI to authenticate and authorize administrator, developer, and end-user operations. CSI has a service provider plug-in model that integrates with the customer's existing security infrastructure.

Unwired Platform does not provide its own security systems for storing and maintaining users and access control rules, but delegates these functions to the enterprise's existing security solutions. Security provider plug-ins for many common security solutions are included with Unwired Platform.

One of the service provider types, the login module, authenticates the user. The login module interface conforms to the Java Authentication and Authorization Service (JAAS). All of the login modules in the Unwired Platform authenticate with userID and password credentials. Multiple login modules — each of which links to a different security store — can be stacked. When the user logs in, each login module attempts authentication in the order specified in the CSI configuration definition. The authentication attempt stops iterating the sequence when authentication has been achieved or rejected.

Security Configurations

Sybase Unwired Platform does not provide proprietary security systems for storing and maintaining users and access control rules, but delegates these functions to the enterprise's existing security solutions.

A security configuration determines the scope of user identity, performs authentication and authorization checks, and can be assigned multiple levels (domain or package). Applications inherit a security configuration when the administrator assigns the application to a domain via a connection template.

Users can be authenticated differently, depending on which security configuration is used. For example, a user identified as "John" may be authenticated different ways, depending on the named security configuration protecting the resource he is accessing: it could be an MBO package, a DCN request, use of Sybase Control Center .

Security configurations aggregate various security mechanisms for protecting Unwired Platform resources under a specific name, which administrators can then assign. Each security configuration consists of:

- A set of configured security providers. Security provider plug-ins for many common security solutions are included with the Sybase Unwired Platform.
- Role mappings (which are set at the domain and package level) that map logical roles to back end physical roles.

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 that accesses data sets 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 repository for:

- Security access policies on the requested resource
- Role memberships

CHAPTER 3 Security Quick Starts, Checklists, and Worksheets

Quick starts are task flows that identify important security setup activities in an Unwired Platform environment. Activities performed by the Unwired Platform administrator, may also require the collaboration or participation of mobile application developers, or Afaria, security, or database administrators, depending on the role distribution of your organization.

To assist you with your quick start activities, use worksheets and checklists as needed.

- Use worksheets to collect and document key decisions relating an activity.
- Use checklists to ensure you have prepared for an activity before starting it.

- *Securing Data at Rest Quick Start*

  Protecting data at the perimeter of a mobile enterprise is insufficieint and ignore a crucial vulnerability — sensitive data stored either on the device or on the runtime data tier are at risk from attackers who only need to find one way inside the network to access this confidential information.
- *Securing Data in Motion Quick Start*

  In a mobile environment, data in motion refers to the transfer of data between the source repository (EIS or backend), and the copies of data from that source as it traverses the perimeter of your organization into mobile networks or the Internet.
- *Securing Access Quick Start*

  Both Unwired Server and Sybase Control Center use Sybase Common Security Infrastructure (CSI). You configure how logins for administrators or devices users are processed.
- *Single Sign-on (SSO) Quick Start*

  Get started with SSO. Perform the activities required by the back-end EIS, using the checklists and worksheets provided for these workflows.

Securing Data at Rest Quick Start

Protecting data at the perimeter of a mobile enterprise is insufficieint and ignore a crucial vulnerability — sensitive data stored either on the device or on the runtime data tier are at risk from attackers who only need to find one way inside the network to access this confidential information.

Because, perimeter defenses like firewalls and Relay Servers cannot protect stored sensitive data from this threat, you must use alternate means to prevent this type of exploitation.

Securing Unwired Platform Data

Secure data managed by the the Unwired Platform data tier. This includes all databases, including those that act as the Unwired Platform synchronization cache.

1. *Securing the Data Infrastructure*

   Secure data by first protecting the infrastructure on which it resides, then securing runtime databases.
2. *Securing Data Tier Databases*

   Secure all databases installed as the Unwired Platform data tier. You can change DBA passwords, grant DBA permissions to other users, and encrypt data and logs.
3. *Encrypting Device Data*

   Encrypting all data on the device client requires multiple techniques.
4. *Securing Sensitive Data On-Device with Data Vault*

   (Not applicable to Hybrid Workflow Container) Developers should use a data vault with device applications to securely store "secrets" on the device. Data vaults are added using the DataVault API.

Encrypting Device Data

Encrypting all data on the device client requires multiple techniques.

Some Unwired Platform components do not support encryption. Review this table to see which components can enable this security feature.

Component	Implementation notes
Device data	Sybase recommend full device encryption with Afaria. See the Afaria documentation for details.
Device client database	(Not applicable to Online Data Proxy) A `<package>DB.generateEncryptionKey()` method in the Object API for MBO packages should always be used during application initialization. It computes a random AES-256 bit encryption key used to encrypt the client database. The encryption key is stored in the data vault.
Data vault	The DataVault APIs provide a secure way to persist and encrypt data on the device. The data vault uses AES-256 symmetric encryption of all its contents. The AES key is computed as a hash of the passcode provided and a "salt" value that can be supplied by the device application developer, or automatically generated through the API.

See also

- *Securing Data Tier Databases* on page 116
- *Registering Applications, Devices, and Users* on page 128

Securing Sensitive Data On-Device with Data Vault

(Not applicable to Hybrid Workflow Container) Developers should use a data vault with device applications to securely store “secrets” on the device. Data vaults are added using the DataVault API.

The data vault holds sensitive artifacts securely, because all data or artifacts in the data vault is strongly encrypted with an AES-256 bit key. Contents can include encryption keys, user and application login credentials, sync profile settings, certificates (as BLOBS).

The data vault requires a password to unlock and access the data from the application. Therefore, a device application must prompt the user to enter this password when the application is opened. Once unlocked, the application can retrieve any other secrets from the vault as needed, all without prompting the user.

Administrators can define a password policy through Sybase Control Center that defines the requirements for an acceptable password. The password policy is stored in the server-side settings database and the client gets those settings when it connects to Unwired Server as part of the settings exchange protocol.

When the client receives the password policy settings, it can populate the settings objects to the data vault. The datavault stores the settings. The client uses the DataVault API to create a vault with a defaut password, set the password policy, and change the password to a password compatible with the policy. If the password is not changed after setting a passwport policy, the application will throw an exception if you then try to access the application or unlock the vault with an incompatible password.

Administrators should discuss the data vault strategy before it is implemented, especially regarding:

- **Failed logins –** Developers can set the number of failed login attempts allowed before the data vault is deleted. Once the vault is deleted the encrypted databases will be un-useable. The application will need to be re-installed, or re-initialized from scratch including deleting the database files to recover
- **Timeouts –** Developers can also set a timeout value so that the data vault locks itself when it's not in use. The user must re-enter the vault password to resume using the application.

For more details about the data vault, see *DataVault* in the developer guide for your application type.

See also

- *Using Login Screens for Data Vaults* on page 131

Data at Rest Security Worksheet

Record information about the data tier and its components. Refer to recorded information to streamline security tasks.

Non-System Administration Password (Installation Defined)

Administration Access	
SQLAnywhere DBA password	

Data Tier Servers

Data Tier Configuration Options	
Separate database and transaction log locations	
Data tier in failover cluster	

Data Tier Server Port Configuration		
Component	**Port**	**Password**
Cache database server		
Cluster database server		
LogData database server		
Messaging database server		n/a

Data TierFailover Clusters

Data Tier Failover Cluster Configuration	
Shared cluster storage path (database files)	
Shared cluster storage path (transaction logs)	
Database server name	

Data Tier Data Paths

Data Tier Database File Locations	
Cache database data path	
Cluster database data path	

Data Tier Database File Locations	
LogData database monitor data path	
LogData database domainlog data path	

Data Tier Transaction Logs

Data Tier Transaction Log File Locations	
Cache database log path	
Cluster database log path	
LogData database monitor log path	
LogData database domainlog log path	

Data Tier Encryption

Data Tier Algorithms Used	
Cache database data and log	
Cluster database data and log	
LogData database monitor data and log	
LogData database domainlog data and log	

Data Tier Backups

Data Tier Backup Policy	
Cache database backup location and frequency	
Cluster database backup location and frequency	
LogData database monitor backup location and frequency	
LogData database domain backup location and frequency	

File System Permissions

Component and Permission Levels	
Data tier permissions	

Data at Rest Security Checklist

Ensure you have secured platform and mobile data that is at rest, either on the corporate LAN or on client devices. Check activities off as you complete them.

Activity	Completed?
Set file system permissions on data tier hosts.	
Secured backup artifacts on data tier hosts.	
Encrypted data and log output for the data tier.	
Encrypted data on the device.	
Ensured that development has enabled a Data Vault for sensitive data.	

Securing Data in Motion Quick Start

In a mobile environment, data in motion refers to the transfer of data between the source repository (EIS or backend), and the copies of data from that source as it traverses the perimeter of your organization into mobile networks or the Internet.

To ensure data is protected along all communication channels, Sybase recommends that you use your existing PKI infrastructure to protect all Unwired Platform communications within and outside your corporate perimeter.

Securing Synchronization

Messaging data is automatically strongly encrypted over HTTP, using a private messaging payload protocol that is completely secure, once established. The message body is a JSON document. Therefore, only replication payloads require administrative intervention.

If your application uses the replication payload protocol, perform steps to secure the replication payload.

1. *Changing Installed Certificates Used for Encryption*

   Unwired Server includes default certificates for all listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.
2. *Modifying Default Synchronization Listener Properties with Production Values*

Once you have determined the degree of secure communication you require, you may need to modify default synchronization listener property values to disable one or more ports or synchronization protocols.

3. *Provisioning Security Artifacts*

   Typically, you must preprovision Unwired Platform security artifacts before applications can be used. The manner by which an artifact is provisioned depends on the artifacts themselves, the device types used, and your deployment environment.
4. *Establishing Encrypted Application Connections*

   Synchronization and messaging connection are encrypted by default. However, for replication connections that use E2EE, the client must be configured correctly to establish connections to the correct HTTP or HTTPS port.

Securing DCNs

The most efficient way to update data from the back-end datasource is to allow the EIS to "push" data changes to Unwired Server. This is accomplished with data change notifications (DCN). DCN can operate over HTTP, however, for a secure communication, use HTTPS and ensure the EIS and Unwired Server use basic authentication.

Note: With HTTPS authentication, the EIS administrator must import the Unwired Sever certificate or CA signing certificate into its truststore.

1. *Preparing SSL Certificates for DCNs*

   DCN, which uses HTTP Basic authentication, uses a Base64-encoded username:password field that can be intercepted by network sniffers. Encrypt DCN communications and always use HTTPS to send DCNs. HTTPS requires the DCN sender import the Unwired Server certificate (or that of its CA signer) into its local equivalent of a truststore.
2. *Creating and Enabling a DCN Security Profile*

   An administrator must enable and configure the HTTPS port for DCN connections as part of a security profile so that developers can construct callouts from the EIS backend to send Unwired Server data change notification (DCN) requests.
3. *Enabling Authorization for Data Change Notification CDB Insertions*

   (Not applicable to Online Data Proxy) All DCN requests are authorized by checking if the user making the request is in the "SUP DCN User" role within the security configuration of the request's target package.

Securing Unwired Platform Runtime Component Communications

There are two different ports that require encryption: the management port for Sybase Control Center (HTTPS), and the management ports used by Unwired Server (IIOPS).

You should also secure the server infrastructure to ensure that runtime binaries for the components performing the communication are protected from internal and external threats.

1. *Securing the Server Infrastructure*

   Before you can secure the runtime, you must first secure the underlying infrastructure. This activity prevents files from being tampered with on the host, also allows the Unwired Server to run with existing security mechanisms.
2. *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners*

   Both Unwired Server and Sybase Control Center include default certificates that are used for these components' HTTPS listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.
3. *Enabling and Configuring Administration Encryption for Unwired Server*

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

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:

      If the security profile authenticates only the server, then only the server must provide a certificate to be accepted or rejected by the client. If the security profile authenticates both the client and the server, then 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.

Profile	Authenticates	Cipher suites
intl	server	• SA_EX-PORT_WITH_RC4_40_MD5 • RSA_EX-PORT_WITH_DES40_CBC_SHA

Profile	Authenticates	Cipher suites
intl_mutual	client/server	• RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA
strong	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
strong_mutual	client/server For example, this is the required option for mutual authentication of Unwired Platform and Gateway.	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
domestic	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • TLS_RSA_WITH_NULL_MD5 • TLS_RSA_WITH_NULL_SHA
domestic_mutual	client/server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • RSA_WITH_NULL_MD5 • RSA_WITH_NULL_SHA

6. Use IIOPS in the Communication Ports subtab by selecting **Secure Management Port** (port 2001), and ensure that Sybase Control Center's Managed Resource properties match. By default, IIOPS is already configured between Unwired Server and Sybase Control Center. .
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.

See also

- *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners* on page 41
- *Securing Multiple Domains* on page 48

Data in Motion Worksheet

Record security setup options for data that moves from one point to another. Refer to recorded information to streamline security tasks.

Runtime Secure Communications: Ports and Certificates

Secure Port Configuration	
Server administration port	
Data change notification port	
Messaging port	
Replication port	
Unwired Server certificates and store location	
Sybase Control Center certificates and store location	
Replication listener server certificate and end-to-end encryption key pairs and store location	

Non-System Administration Password (Installation Defined)

Administration access	
Windows cluster administrator password	

File System Permissions

Component and Permission Levels	
Unwired Server host permissions	

Production Grade Security Providers for Administration

These providers enable role-based accesscontrol (RBAC) for administrators.

Providers for Administration Access	
Provider type	
Users or Groups for platform admin role mapping	
Users or Groups for domain admins role mapping	

Domains and Tenants

Domains and Tenancy Strategy	
Numbers of domains needed	
Names of domains	
Domain strategy employed	
Domain administrators assigned to each domain	
Security configurations assigned to each domain	

DCN Security

DCN SSL security	
Security profile name	
Authentication strength	
Certificate names and locations	
Security configuration	
SUP DCN User role mapping	

Securing Access Quick Start

Both Unwired Server and Sybase Control Center use Sybase Common Security Infrastructure (CSI). You configure how logins for administrators or devices users are processed.

CSI security providers perform these functions:

- **Authentication –** is performed using JAAS style LoginModules.
- **Authorization –** follows a role-based access control model.
- **Audit –** keeps an audit trail of authentication/authorization decisions made by CSI.
- **Role mapping –** when logical roles are used, allows you to map physical roles to logical ones.

Enabling Logins for Unwired Platform

Logins are configured differently for administrators and devices users. Sybase recommends that you keep the security configuration used by MBO packages separate from the “admin” configuration used by Unwired Platform administrators (platform administrators or domain administrators).

Therefore, determine your tenancy and domain strategy before configuring logins for administrators and users. With domains created, you can then easily assign security configurations or to the appropriate domain.

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. *Enabling Authentication and RBAC for Administrator Logins*

   Role based access control (RBAC) for administrators is always performed by Unwired Server: Sybase Control Center automatically delegates administrator authentication to the providers configured for the "admin" security configuration on the "default" domain. When you install Unwired Platform, only the PreconfiguredUserLogin module is used, so you must initially log in with the administrator credentials defined with the installer.
3. *Enabling Authentication and RBAC for User Logins*

   Enable authentication and role-based access control (RBAC) for device user logins by creating a new security configuration (that is, one that is distinct from the "admin" security configuration on the "default" domain), and mapping roles, then assigning it.

Single Sign-on (SSO) Quick Start

Get started with SSO. Perform the activities required by the back-end EIS, using the checklists and worksheets provided for these workflows.

Enabling Single Sign-on for DOE-C Packages

Enable single sign-on (SSO) over secure paths for Data Orchestration Engine Connector (DOE-C) packages.

1. *Preparing Your SAP Environment for Single Sign-on*

   Verify that the SAP® enterprise information system (EIS) is configured correctly to accept SSO connections from Unwired Server.
2. *Configuring X.509 Certificates for SAP Single Sign-on*

   Import, export, and generate the X.509 certificates that secure communication paths between Unwired Server and the SAP enterprise information system (EIS), and for client authentication, including single sign-on (SSO) with X.509 or SSO2 tokens.
3. *Deploying and Configuring DOE-C Packages*

   Unlike Mobile Workflow or MBO packages that use Sybase Control Center to deploy packages to Unwired Server, you must deploy the DOE-C package to specific domain using the DOE-C command line utility (CLU). Once deployed, the DOE-C package is visible and manageable from Sybase Control Center.
4. *Creating Security Profiles to Enable Mutual Authentication for SAP*

   Create security profiles and associate them with X.509 server certificates that can be used to establish secure connections between a client, Unwired Server, and the SAP EIS.
5. *Enabling the HTTPS Port and Assigning the Unwired Server Security profile*

   Enable an HTTPS port for secure communication between Unwired Server and the SAP EIS.
6. *Enabling the DOE-C Connection*

   Configure the SAP Data Orchestration Engine Connector (DOE-C) connection pool between Unwired Server and the SAP EIS. This is the port on which Unwired Server communicates with the DOE, including forwarding subscriptions and allowing client operations to flow through to the DOE.
7. *Security Configurations That Implement Single Sign-on Authentication*

   Use the CertificateAuthenticationLoginModule authentication module to implement X. 509 authentication or HttpAuthenticationLoginModule to implement SSO2.
8. *Provisioning with Unwired Server*

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

See also

- *SAP Single Sign-on and DOE-C Package Overview* on page 75
- *Single Sign-on Authentication* on page 70

Enabling Single Sign-on for OData Applications

Enable single sign-on (SSO) over secure paths for OData applications.

1. *Preparing the SAP Gateway*

   Configure the SAP Gateway to push OData application data to Unwired Server, including configuring the RFC destination for HTTPS on the Gateway.
2. *Preparing Your SAP Environment for Single Sign-on*

   Verify that the SAP® enterprise information system (EIS) is configured correctly to accept SSO connections from Unwired Server.
3. *Using Keytool to Generate Self-Signed Certificates and Keys*

   Whenever possible, use a PKI system and a trusted CA to generate production-ready certificates and keys that encrypt communication among different Unwired Platform components. You can then use keytool to import and export certificate to the platform's keystores and truststores. Otherwise, you can also use keytool to generate self-signed certificates and keys.
4. *Configuring X.509 Certificates for SAP Single Sign-on*

   Import, export, and generate the X.509 certificates that secure communication paths between Unwired Server and the SAP enterprise information system (EIS), and for client authentication, including single sign-on (SSO) with X.509 or SSO2 tokens.
5. *Creating Security Profiles to Enable Mutual Authentication for SAP*

   Create security profiles and associate them with X.509 server certificates that can be used to establish secure connections between a client, Unwired Server, and the SAP EIS.
6. *Enabling the HTTPS Port and Assigning the Unwired Server Security profile*

   Enable an HTTPS port for secure communication between Unwired Server and the SAP EIS.
7. *Security Configurations That Implement Single Sign-on Authentication*

   Use the CertificateAuthenticationLoginModule authentication module to implement X. 509 authentication or HttpAuthenticationLoginModule to implement SSO2.
8. *Provisioning Security Artifacts*

   Typically, you must preprovision Unwired Platform security artifacts before applications can be used. The manner by which an artifact is provisioned depends on the artifacts themselves, the device types used, and your deployment environment.

See also

- *SAP Single Sign-on and Online Data Proxy Overview* on page 89
- *Single Sign-on Authentication* on page 70

Enabling Single Sign-on for Mobile Business Object Packages

Enable single sign-on (SSO) over secure paths for mobile business object (MBO) packages.

1. *Single Sign-on for SAP MBO Package Prerequisites*

   Before implementing SSO for SAP MBO packages, configure the MBOs so client credentials can be propagated to the EIS and, if enabling SSO for a Workflow application, add the appropriate starting point.
2. *Preparing Your SAP Environment for Single Sign-on*

   Verify that the SAP® enterprise information system (EIS) is configured correctly to accept SSO connections from Unwired Server.
3. *Installing the SAPCAR Utility*

   Unzip and install the latest SAPCAR utility on your Unwired Server or Unwired WorkSpace host, which you can use to extract the contents of compressed SAP files, for example RFC and cryptographic library files.
4. *Installing the SAP Cryptographic Libraries on Unwired Platform*

   Installation and configuration is required if you want to configure Secure Network Communications (SNC) for Unwired Platform SAP JCo connections. SNC may be required by the SAP EIS in question, if SSO2 tokens or X.509 certificates are used for connection authentication.
5. *Configuring X.509 Certificates for SAP Single Sign-on*

   Import, export, and generate the X.509 certificates that secure communication paths between Unwired Server and the SAP enterprise information system (EIS), and for client authentication, including single sign-on (SSO) with X.509 or SSO2 tokens.
6. *Creating Connections and Connection Templates*

   Create a new connection or connection template that defines the properties needed to connect to a new data source.
7. *Security Configurations That Implement Single Sign-on Authentication*

   Use the CertificateAuthenticationLoginModule authentication module to implement X.509 authentication or HttpAuthenticationLoginModule to implement SSO2.
8. *Provisioning Security Artifacts*

   Typically, you must preprovision Unwired Platform security artifacts before applications can be used. The manner by which an artifact is provisioned depends on the artifacts themselves, the device types used, and your deployment environment.
9. *Single Sign-on for SAP MBO Package Postrequisites*

After configuring SSO for SAP MBO packages on Unwired Server, install certificates on the mobile device and test them.

See also

- *SAP Single Sign-on and Mobile Business Object Package Overview* on page 77
- *Single Sign-on Authentication* on page 70

SSO Worksheet

Record SSO setup options and refer to recorded information to streamline SSO setup tasks.

SAP SSO Ports

Secure Port Configuration	
ICM HTTPS port	
SAP Gateway HTTPS port	

SAP Certificate Values

X.509 Certificate Properties	
User ID (DN) mapped to AS ABAP	
Certificate Alias	
Authentication strength	

SAP Connector Properties

SAP Connector Configuration	
Package deployment domain	
SNC certificate file	
SNC library location	

SSO Checklists

Mark activities you complete to ensure you have performed security tasks for SSO.

SAP MBO Checklist

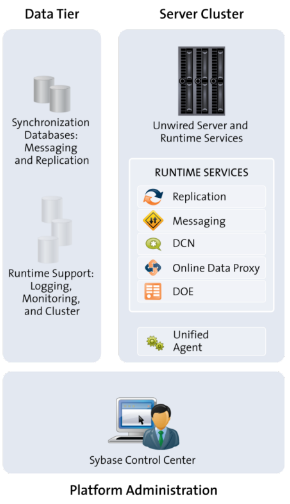
Activity	Completed?
Validate client IDs exist in SAP security repositories.	
(OData) Prepare the SAP Gateway.	

Activity	Completed?
(MBO) Validate that MBO package uses a JCo connector.	
(MBO) Validate that SAP function modules are exposed as Web services.	
(X.509) Use PKI to create certificates and keys for SSO.	
Enable SAP systems to communicate with Unwired Server (using either SSO2 tokens or X.509 certificates).	
Install required utilities and libraries onto Unwired Server hosts.	
Import SSL encryption certificates into Unwired Server stores.	
Create a security profile and enable the HTTPS port.	
(X.509) Import SSO certificates into Unwired Server stores.	
Create application templates to connect to the SAP data source (DOE, JCo/SNC for OData, or Web Services for MBO).	
Create security configurations and assign to the required packages/domains.	
Deploy packages to required domains.	
Provision devices with certificates.	
Validate and test connections to SAP backends.	

CHAPTER 4 Server Security

The Unwired Server provides data services to device clients by interacting with the data tier. The data tier is installed along with server tier components, to the internal corporate LAN.

Each runtime service uses its own communication port (secured and unsecured).



Secure the server runtime by performing activities that secure the infrastructure and administration of those components, in addition to enabling user authentication and secure communication.

1. *Securing the Server Infrastructure*

   Before you can secure the runtime, you must first secure the underlying infrastructure. This activity prevents files from being tampered with on the host, also allows the Unwired Server to run with existing security mechanisms.
2. *Securing Platform Administration*

   Use the Web-based console called Sybase Control Center to remotely and securely administer Unwired Platform.
3. *Enabling Authentication and RBAC for User Logins*

   Enable authentication and role-based access control (RBAC) for device user logins by creating a new security configuration (that is, one that is distinct from the "admin" security configuration on the "default" domain), and mapping roles, then assigning it.
4. *Encrypting Synchronization for Replication Payloads*

(Not applicable to Online Data Proxy) By default, the Unwired Server replication listener is configured to use TLS for end-to-end encryption (E2EE) on HTTP and HTTPS ports, and SSL for encryption on HTTPS ports.

5. *Encrypting Other Listeners for Unwired Server*

   By default all other Unwired Platfrom listeners are encrypted using SSL. However, if you need to modify this configuration, review these steps.

Securing the Server Infrastructure

Before you can secure the runtime, you must first secure the underlying infrastructure. This activity prevents files from being tampered with on the host, also allows the Unwired Server to run with existing security mechanisms.

1. *Handling Intrusion Detection/Prevention Software*

   A personal firewall, or intrusion detection/prevention software (IPS or IDPS), can cause Unwired Platform components to malfunction or not function at all. Unwired Platform uses regular IP communication between components on the primary network interface of a computer, even when all components are installed on the same host.
2. *Setting File System Permissions*

   Unwired Platform runs as a collection of Windows services. During installation, you are prompted with a Logon as request. The credentials collected are then used to run the service under that account. In a cluster installation, the same Windows user would be configured for all installations and respective Window services that are subsequently installed.

See also

- *Securing Platform Administration* on page 32

Handling Intrusion Detection/Prevention Software

A personal firewall, or intrusion detection/prevention software (IPS or IDPS), can cause Unwired Platform components to malfunction or not function at all. Unwired Platform uses regular IP communication between components on the primary network interface of a computer, even when all components are installed on the same host.

If the local network interface is secured by intrusion detection/prevention software (IPS or IDPS, for example, McAfee Host Intrusion Prevention software or equivalent), you must configurethe security software to allow all network communication between Unwired Platform components.

For a single-node installation of all of the Sybase Unwired Platform components, try one of these options to work around the limitations imposed by the host intrusion prevention software and policy settings, without violating any security policy, until the settings of your security software are adjusted to the needs of Unwired Platform.

Choose an option:

- Removing the host machine from the network – this option ensures that all interconnections between Sybase Unwired Platform components are treated as local traffic and is not be flagged as incoming connections from external sources, thereby causing connection failures due to security policy setting. This option is suitable when you use your laptop in a network other than your corporate network, and want to demonstrate a mobile solution using a simulator or emulator with all components running on the same machine. To use this option:
  1. Stop the Sybase Unwired Platform services in the correct order. See *Starting and Stopping Unwired Platform Server Services* in *System Administration*.
  2. Disconnect the host from all networks.
  3. Restart Sybase Unwired Platform services in the correct order.
  4. Change the Sybase Control Center URL link to use "localhost" or <*yourhostname*> as the host name, instead of the original fully qualified host name of the machine that included the domain name (for example: `https://localhost:8283/scc`, or `https://`*`yourhostname`*`:8283/scc`). Accept any security warnings to connect to Sybase Control Center.
- Connecting the host to the corporate network – this option ensures that all interconnections among Sybase Unwired Platform components are internal to your corporate network and validated against the corporate network security policy. The option of connecting to corporate network through VPN is especially suitable when you use your laptop in a network other than your corporate network, and want to demonstrate a mobile solution using your physical devices, and need outgoing connections to a backend Enterprise Information System (EIS) or Relay Server (Sybase Hosted Relay Server or otherwise).
  1. Stop the Sybase Unwired Platform services in the correct order. See the *Starting and Stopping Unwired Platform Server Services* topic in the *System Administration*.
  2. Reconnect the host to your corporate network directly or through corporate VPN, to ensure that the corporate network security policy applies.
  3. Restart Sybase Unwired Platform services in the correct order.
  4. Change the Sybase Control Center URL link to use "localhost" or <*yourhostname*> as the host name, instead of the original fully qualified host name of the machine that included the domain name (for example: `https://localhost:8283/scc`, or `https://`*`yourhostname`*`:8283/scc`). Accept any security warnings to connect to Sybase Control Center.
- Configuring the firewall software to allow connections to the ports the Unwired Platform uses. For a list of ports, see *Unwired Platform Ports* in *System Administration*.

Always check for the latest available patches and updates for your Unwired Server version on *http://downloads.sybase.com/swd/base.do?client=support*.

Setting File System Permissions

Unwired Platform runs as a collection of Windows services. During installation, you are prompted with a **Logon as** request. The credentials collected are then used to run the service under that account. In a cluster installation, the same Windows user would be configured for all installations and respective Window services that are subsequently installed.

You can restrict permissions after installation by removing most users and groups from the Unwired Platform installation directory.

1. Open File Explorer.
2. Right-click *`<UnwiredPlatform_InstallDir>`*, and click **Properties**.
3. On the **Security** tab, click **Advanced**.
4. Unselect **Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.**.
5. In the confirmation pop-up, choose **Copy**, then select **Replace permission entries on all child objects with entries shown here that apply to child objects.**.
6. In the table of Permission entries, remove all users except the user account that was configured as the Logon user for the Windows services. If another user is responsible for some activities extend the necessary permissions to this administrator. For example, if the individual is only reading log files, you may choose to limit permissions to read only.
7. Click **OK**.

Securing Platform Administration

Use the Web-based console called Sybase Control Center to remotely and securely administer Unwired Platform.

Sybase Control Center relies on a Windows service called Sybase Control Center X.X that runs on each Unwired Server on the cluster. The service handles communication between Sybase Control Center and Unwired Server runtime components.

1. *Enabling Authentication and RBAC for Administrator Logins*

   Role based access control (RBAC) for administrators is always performed by Unwired Server: Sybase Control Center automatically delegates administrator authentication to the providers configured for the "admin" security configuration on the "default" domain. When you install Unwired Platform, only the PreconfiguredUserLogin module is used, so you must initially log in with the administrator credentials defined with the installer.
2. *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners*

   Both Unwired Server and Sybase Control Center include default certificates that are used for these components' HTTPS listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.
3. *Enabling and Configuring Administration Encryption for Unwired Server*

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

4. *Securing Multiple Domains*

   To prevent role mapping leaks between multiple tenant domains, configure domains and assign shared security configurations.

See also

- *Securing the Server Infrastructure* on page 30
- *Enabling Authentication and RBAC for User Logins* on page 51

Enabling Authentication and RBAC for Administrator Logins

Role based access control (RBAC) for administrators is always performed by Unwired Server: Sybase Control Center automatically delegates administrator authentication to the providers configured for the "admin" security configuration on the "default" domain. When you install Unwired Platform, only the PreconfiguredUserLogin module is used, so you must initially log in with the administrator credentials defined with the installer.

The PreconfiguredUserLoginModule does not enforce password strength or change policies that would typically be in place for a production environment. Therefore, substitute the PreconfiguredUserLogin module with one that is suitable for a production environment. Subsequent logins are then performed with user credentials assigned to the platform or domain administrator role.

1. *Logging Into Sybase Control Center with an Installer-Defined Password*

   The person acting as platform administrator logs in to Sybase Control Center for the first time after installation.
2. *Making "Admin" Security Configuration Production-Ready*

   Replace the default PreConfiguredUserLoginModule with new production-ready providers.
3. *Disabling Authentication Caching and Increasing Log Levels*

   Temporarily disable administrator authentication, so the new provider can be validated. Also increase log levels to capture more detailed events in case you need to troubleshoot problems.
4. *Validating the Production "Admin" Security Configuration*

   Once LDAP has been added as a provider to both the "admin" security configuration for Unwired Server and the CSI property file for Sybase Control Center you can test the login before removing the PreconfiguredUser login module from both components' configurations.
5. *Enabling Authentication Caching and Reducing Log Levels*

   Re-enable administrator authentication as required by your environment, and reduce log levels to a value more appropriate for normal security operations.

See also

- *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners* on page 41
- *SUP Administrator and SUP Domain Administrator* on page 65
- *Authentication in Unwired Platform* on page 53
- *Delegated Login Security Provider* on page 38
- *Preconfigured User Login Security Provider* on page 36

Logging Into Sybase Control Center with an Installer-Defined Password

The person acting as platform administrator logs in to Sybase Control Center for the first time after installation.

During installation, the person installing Unwired Platform defines a password for the supAdmin user. This password is used to configure the Preconfigured login module that performs the administrator authentication.

Note: This installer-defined password is not intended to be a permanent administrator credential. You must replace this module with a production-grade authentication module, typically LDAP.

---

1. Launch Sybase Control Center.
2. Enter `supAdmin` for the user name and type the <*`supAdminPwd`*> for the password.
3. Click **Login.**
4. Open the Unwired Platform perspective and authenticate with Unwired Server using the same credentials used to log into Sybase Control Center.

Making "Admin" Security Configuration Production-Ready

Replace the default PreConfiguredUserLoginModule with new production-ready providers.

Note: Please note these restrictions before beginning:

- For LDAPLoginModule, special characters (for example, `, = : ' " * ? &`) cannot be used in the user name defined with the Authentication Filter property. This same property also does not support Chinese or Japanese characters in the user name and password properties.
- For PreConfiguredUserLoginModule, the User Name property also cannot contain the same special characters (that is, `, = : ' " * ? &`).

See also

- *Disabling Authentication Caching and Increasing Log Levels* on page 40

Adding a Production-Grade Provider

Modify the "admin" security configuration to add a production-grade provider, typically an LDAPLoginModule. Most companies use an LDAP directory to maintain internal user accounts. This module integrates with most LDAP servers including Active Directory.

Prerequisites

Ensure you have gathered information on the provider you will be using. See *Gathering Provider Information*.

Task

Configure the "admin" security configuration on the "default" domain to authenticate only platform and domain administrators. Do not use this security configuration for MBO packages and mobile users.

1. In the navigation pane of Sybase Control Center, expand the **Security** folder, then click the security configuration named **admin**.
2. In the administration pane, click the **Authentication** tab.
3. Add an LDAPLoginModule, configuring the providerURL, serverType, bind user, bind password, search base, and other properties determined by you and the LDAP administrator.
4. Add a `ControlFlag` attribute for the configured LDAP login module, and set the value to `sufficient`.
5. Make the LDAP module the first module in the list.

   Note: Do not remove the PreConfiguredUser login module from the list of login modules used by the "admin" security configuration until the LDAP login module has been tested.
6. Select the **General** tab, select **Validate**, then **Apply**.
7. Click **OK**.

See also

- *Preconfigured User Login Security Provider* on page 36
- *Preconfigured User Authentication Properties* on page 171
- *LDAP Security Provider* on page 37
- *LDAP Configuration Properties* on page 151

Gathering Provider Information

Production environments rely on a production-grade security provider (commonly an LDAP directory) to authenticate administrators. To identify the changes you may need to effect for Unwired Platform use, first understand how the provider is structured and organized.

Consider which users need to be in the SUP Administrator or SUP Domain Administrator role, then identify or create groups in your provider that corresponding to these roles. You must also allocate a group for the DCN User role.

Note: If you have installed an earlier version of Unwired Platform as part of a development deployment, you may have an OpenDS LDAP server running in your environment, and both Unwired Platform and Sybase Control Center may be using this directory. Sybase no longer uses this directory and strongly encourages you to use a different LDAP directory.

1. Evaluate existing groups.

   If there are existing groups that seem to already contain the right subjects that correspond to SUP DCN User, SUP Administrator, and SUP Domain Administrator platform roles, you can use those groups. The names need not be exact, as you can map them in Sybase Control Center to address any differences.
2. If no sufficient group exists, add them for Unwired Platform.
3. Add subjects to these groups to assign Unwired Platform corresponding permissions.
4. Determine what values are needed for the login module properties in Unwired Platform.
   For example, for an LDAP login module you need values for the providerURL, serverType, bind user, bind password, search base and so on.

See also

- *Built-in Security Providers for "Admin" Security Configuration* on page 36

Built-in Security Providers for "Admin" Security Configuration

Unwired Server supports a variety of built-in security providers that authenticate administrators users using the "admin" security configuration on the "default" domain. Administrators define one or more security providers to replace the default Preconfigured User login module.

You can configure a provider of a given type, only if that provider is available on the enterprise network.

See also

- *Gathering Provider Information* on page 36

Preconfigured User Login Security Provider

Preconfigured login is configured to authenticate the supAdmin user with a password that was defined when Unwired Platform was installed. Therefore, an administrator must use `supAdmin` with <*supAdminPwd*> when initially logging in to Sybase Control Center for the first time.

Note: Do not forget this installer-defined password. The installer hashes the password with a SHA-256 algorithm before it is saved as part of the PreconfiguredLoginModule configuration, and it cannot be returned to clear text once it is hashed.

Once logged in, the Unwired Platform administrator immediately reconfigures the "admin" security configuration to replace this provider with a production-grade security provider like LDAP. If you configure a new provider in Unwired Platform and Sybase Control Center and login fails, review possible login failure solutions in the *Troubleshooting* guide.

See also

- *Preconfigured User Authentication Properties* on page 171
- *Adding a Production-Grade Provider* on page 35
- *Enabling Authentication and RBAC for Administrator Logins* on page 33

LDAP Security Provider

(Not applicable to Online Data Proxy) The LDAP security provider includes authentication, attribution, and authorization providers. Add an LDAP provider to a security configuration to authenticate administrator logins (on the "admin' security configuration on the "default" domain) or device user logins (any custom security configuration for that purpose).

You can configure these providers:

- **LDAPLoginModule**– provides authentication services. Through appropriate configuration, you can enable certificate authentication in **LDAPLoginModule**.
- (Optional)**LDAPAuthorizer** or **RoleCheckAuthorizer** – provide authorization services for **LDAPLoginModule**. **LDAPLoginModule** works with either authorizer. In most production deployments, you must always 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**.

  The RoleCheckAuthorizer is used with every security configuration but does not appear in Sybase Control Center.

  Use **LDAPAuthorizer** only when **LDAPLoginModule** is not used to perform authentication, but roles are still required to perform authorization checks against the LDAP data store. If you use **LDAPAuthorizer**, always explicitly configure properties; for it cannot share the configuration options specified for the **LDAPLoginModule**.

You need not enable all LDAP providers. You can also implement some LDAP providers with providers of other types. If you are stacking multiple LDAP providers, be aware of, and understand the configuration implications.

See also

- *LDAP Configuration Properties* on page 151

- *Adding a Production-Grade Provider* on page 35
- *Stacking Providers and Combining Authentication Results* on page 98

Delegated Login Security Provider

Delegated login module is used only to authenticate administrators from Sybase Control Center. It allows Sybase Control Center to delegate authentication and authorization of administrators to Unwired Server.

- On a new installation of Unwired Platform, this provider is automatically configured. `roles-map.xml` and `csi.properties` are configured accordingly, and delegation to Unwired Server for authentication and role mapping automatically occurs.
- On an upgrade installation, the Sybase Control Center configuration from an earlier version is retained. However, Sybase strongly recommends that you change the Sybase Control Center configuration using `csi.properties` and `roles-map.xml` to use only this Delegated module. See *Replacing Existing Sybase Control Center Login Modules with Delegated Login Module* in the *Release Bulletin*.

See also

- *Role Mapping (roles-map.xml) Configuration File* on page 149
- *CSI Configuration (csi.properties) for Sybase Control Center* on page 150
- *Enabling Authentication and RBAC for Administrator Logins* on page 33

Mapping Unwired Platform Logical Roles to Physical Roles

Unwired Platform requires that you map these default platform roles: SUP Administrator and SUP Domain Administrator. In a development environment, these mappings are automatic if the provider roles use the exact logical role names. If the development environment roles do not match, then you must manually map them.

Use Sybase Control Center to map these logical roles to the appropriate physical roles or groups in the underlying security provider. The mapping determines 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.

1. Open Sybase Control Center.
2. In the left navigation pane, expand **Domains**.
3. Expand the **Default** domain.
4. Open the Security folder and click the **Admin** security configuration.
5. Map roles to the security provider groups or roles:
   - If default roles exactly match the names in the security provider repository, select **AUTO**.
   - If the default role differ from those manually added to the repository, click the list adjacent to the logical role and choose **Map Role**. The Role Mappings dialog allows

you to manually set logical and physical role mappings.Once saved, the state automatically changes to MAPPED.

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

Encrypting a Sensitive Property in CSI.PROPERTIES

Generate an encrypted string to use as the sensitive property value in the csi.properties configuration file.

1. Run the following command to generate the encrypted string, changing the path according to your Unwired Platform installation home (the example assumes default location C:\Sybase\UnwiredPlatform).`java -jar C:\Sybase\UnwiredPlatform\Servers\UnwiredServer\lib\ext\csi-tool.jar csi.encmessage @C:\Sybase\UnwiredPlatform\Servers\UnwiredServer\Repository\CSI\csibootstrap.properties --text secret`
   Example Output:

```
CSITool 4.2 (c) Copyright 2005-2007 Sybase, Inc. (4.2M6/2011/09/08
11:19:05 PDT)


Running task: csi.encmessage

Successfully encrypted message.  The output was sent to stdout.
1-AAAAEgQQ9nMbR0ho3R1a6n+g8IfI1MiRr2tf48byz8GOzKLx/
raDbB62yW7Tevagbeuez7wTfuVK8U
DUUZUWhJHdTLq4yWu1Wpk1HvxChNcJpfHmHSc=s
```

2. Edit `C:\Sybase\SCC-3_2\conf\csi.properties` and update **sensitive**.

   For example, "CSI.loginModule.XX.options.BindPassword.e" for the SUP LDAP login module) with the encrypted string generated in the previous step. Replace "XX" with the index assigned to the login module entry in your csi.properties. The suffix ".e" denotes the property value is encrypted.
3. Edit `C:\Sybase\SCC-3_2\bin\scc.properties` and add `jvmopt=-Dcom.sybase.security.BootstrapConfigurationURL=file:///C:/Sybase/UnwiredPlatform/Servers/UnwiredServer/Repository/CSI/csibootstrap.properties`
4. Copy the keystore file `C:\Sybase\UnwiredPlatform\Servers\UnwiredServer\Repository\CSI\csikeystore.jceks` to `C:\Sybase\SCC-3_2`.
5. Restart SCC. Verify agent.log to ensure no errors occurred while decrypting the password.

   Repeat this step for all SCC nodes that require password encryption.

Disabling Authentication Caching and Increasing Log Levels

Temporarily disable administrator authentication, so the new provider can be validated. Also increase log levels to capture more detailed events in case you need to troubleshoot problems.

1. Disable authentication:
   a) In the left navigation pane, expand the **Security** folder.
   b) Select the **Admin** security configuration, and display its properties.
   c) In the right administration pane, select the **Settings** tab.
   d) Set the cache timeout value to 0, which tells Unwired Server to not cache results.
2. Increase log levels to a more sensitive value:
   a) In the left navigation pane, expand the **Servers** folder and select the server to configure.
   b) Select **Log**.
   c) In the right administration pane, click the **Settings** tab.
   d) For the security log component, set the log level to DEBUG, which provides detailed system information, warnings, and all errors.

See also

- *Making "Admin" Security Configuration Production-Ready* on page 34
- *Authentication Cache Timeouts* on page 97

Validating the Production "Admin" Security Configuration

Once LDAP has been added as a provider to both the "admin" security configuration for Unwired Server and the CSI property file for Sybase Control Center you can test the login before removing the PreconfiguredUser login module from both components' configurations.

1. Log in to Sybase Control Center using the login values of an LDAP user that is in an LDAP group mapped to the "SUP Administrator" logical role.
2. If the login succeeds:
   a) Remove PreconfiguredUser login module from Unwired Server and Sybase Control Center setup locations.
   b) Reduce the logging levels for both Unwired Server and Sybase Control Center to Info or Warn.
   a) Restart Unwired Server.
3. If the login fails:
   a) Check Sybase Control Center's log in `<`*`UnwiredPlatform_InstallDir`*`>\SCC-`*`X_X`*`\log\agent.log` to see if authentication failed with this component.
   b) If no issues are identified, continue checking with the Unwired Server log in `<`*`UnwiredPlatform_InstallDir`*`>\Servers\UnwiredServer\logs\<`*`ClusterName`*`>-server.log`.

c) If no issues are immediately apparent, review security issues documented in *Troubleshooting* for possible resolution guidelines.

Enabling Authentication Caching and Reducing Log Levels

Re-enable administrator authentication as required by your environment, and reduce log levels to a value more appropriate for normal security operations.

1. Enable authentication:

a) In the left navigation pane, expand the **Security** folder.

b) Select the **Admin** security configuration, and display its properties.

c) In the right administration pane, select the Settings tab.

d) Set the cache timeout value in seconds. The default is 3600 seconds.

The **Authentication cache timeout** determines how long authentication results should be cached before the administrator is required to reauthenticate.

2. Return log levels to a less sensitive value:

a) In the left navigation pane, expand the **Servers** folder and select the server to configure.

b) Select **Log**.

c) In the right administration pane, click the **Settings** tab.

d) For the security log component, set the log level to WARN.

See also

- *Authentication Cache Timeouts* on page 97

Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners

Both Unwired Server and Sybase Control Center include default certificates that are used for these components' HTTPS listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.

Prerequisites

By default, Unwired Server includes two security profiles, which is used by secure management and Data Change Notification (DCN) listeners: default and default_mutual. Therefore, you need to determine what type of authentication is required. The security profile you use determines which certificate file you need, and where they need to be deployed. The most secure profile is default_mutual, whereby components are mutually authenticated.

- The default security profile uses domestic authentication and uses the alias of "sample1". With this authentication type, Unwired Server sends its certificate to the client (that is, either Sybase Control Center or DCNs). However, it does not require a certificate in return

from the client. Instead, you must configure the client to trust the Unwired Server certificate.
- The default_mutual security profile uses domestic_mutual authentication and uses the alias of "sample2". This authentication type requires that Sybase Control Center and Unwired Server truststores each contain a copy of the other component's certificate.

For details about what cipher suites are supported for domestic and domestic_mutual authentication, see *Creating an SSL Security Profile in Sybase Control Center* in the Sybase Control Center online help.

Task

Note: Because secure DCN has automatically been configured to use these same profiles by default, you are updating certificates used for secure DCN communication. If you want DCN to use a unique profile and certificates, see *EIS Tier Security*.

1. Generate new production-ready certificates:
   a) For Unwired Server: if you are using default , create new server certificates for Unwired Server and keep the current alias of "sample1"; if you are using default_mutual also generate new server certificates for Sybase Control Center and keep the current alias of "sample2". This replaces the sample certificates in this keystore.
      - If you use a PKI system, ensure that the generated certificates and key pairs are signed by the Certificate Authority (CA) certificate that is widely trusted in your organization. Unwired Platform is compliant with certificates and key pairs generated from most well known PKI systems. Sybase recommends that you use this option.
      - If you do not use a PKI system, use the **keytool** utility to generate new self-signed certificates by following these steps. For an example of a **keytool** command, see *Preparing Certificates and Key Pairs*.

      Note: For a clustered environment, set the CN of the certificate to `*.`*`MyDomain`*. The truststore and keystore files, as well as the definitions for default and default_mutual profiles are then synchronized across the cluster. As a result, there will only ever be a single certificate shared by all nodes that are members of the same cluster.

   b) For Sybase Control Center: generate a new certificate for this keystore with a "jetty" alias. This replaces the default self-signed certificate installed in that keystore.
2. Import production-ready certificates, then update the security profile to associate these files with the Unwired Server encrypted port.
   a) Use **keytool** to import the new production certificates into the primary Unwired Server keystore.
   b) In the left navigation pane, expand the **Servers** folder and select the primary Unwired Server.

   c) Select **Server Configuration**.
   d) In the right administration pane, click **General** then **SSL Configuration**.
   e) Optional. If you have used a different alias, rather than keep the alias of "sample1", locate the profile name row and modify the alias name to match the one used by your certificate.
   f) Optional. If you are using a PKI system that includes OCSP, configure an OCSP responder. See *Enabling OCSP*.
3. Update Sybase Control Center keystores and configure it to also use these production-ready certificates.
   a) Use **keytool** to import the new production Unwired Server certificate into the Sybase Control Center keystore at `<UnwiredPlatform_InstallDir>\SCC-XX\plugins\com.sybase.supadminplugin_X.X.X\security\truststore.jks`.
   b) Open `<UnwiredPlatform_InstallDir>\SCC-XX\services\Messaging\lib\eas\lib\Repository\Server\EmbeddedJMS\Instance\com\sybase\djc\server\ApplicationServer\EmbeddedJMS.properties`, and revise the *filePath*,*keyStoreName*, *trustStoreName* and *password* properties, so that Sybase Control Center can locate and access these stores.
4. Optional. If you are using default_mutual authentication, use **keytool** to import the new server certificate for Sybase Control Center into the primary Unwired Server truststore.
5. Replace the default certificate for Sybase Control Center's HTTPS listener. Use **keytool** to import the new Sybase Control Center certificate with the "jetty" alias to the `<UnwiredPlatform_InstallDir>\SCC-X_X\keystore` keystore.

See also

- *Enabling Authentication and RBAC for Administrator Logins* on page 33
- *Enabling and Configuring Administration Encryption for Unwired Server* on page 18

Enabling OCSP

(Optional) Enable OCSP (Online Certificate Status Protocol) to determine the status of a certificate used to authenticate a subject: current, expired, or unknown. OCSP configuration is enabled as part of server level SSL configuration. OCSP checking must be enabled if you are using the CertificateAuthenticationLoginModule and have set Enable revocation checking to true.

Enable OCSP for an Unwired Server when configuring SSL.

1. To enable OCSP when doing certificate revocation checking, check **Enable OCSP**.
2. Configure the responder properties (location and certificate information):

Responder Property	Details
URL	A URL to responder, including its port. For example, `https://ocsp.example.net:80`.
Certificate subject name	The subject name of the responder's certificate. By default, the certificate of the OCSP responder is that of the issuer of the certificate being validated. Its value is a string distinguished name (defined in RFC 2253), which identifies a certificate in the set of certificates supplied during cert path validation. If the subject name alone is not sufficient to uniquely identify the certificate, the subject value and serial number properties must be used instead. When the certificate subject name is set, the certificate issuer name and certificate serial number are ignored. For example, `CN=MyEnterprise, O=XYZCorp`.
Certificate issuer name	The issuer name of the responder certificate. For example, `CN=OCSP Responder, O=XYZCorp`.
Certificate serial number	The serial number of the responder certificate.

See also

- *Creating an SSL Security Profile in Sybase Control Center* on page 110

Using Keytool to Generate Self-Signed Certificates and Keys

Whenever possible, use a PKI system and a trusted CA to generate production-ready certificates and keys that encrypt communication among different Unwired Platform components. You can then use **keytool** to import and export certificate to the platform's keystores and truststores. Otherwise, you can also use **keytool** to generate self-signed certificates and keys.

Review sample commands, to see how to use **keytool** to import, export, and generate certificates and keys:

1. If you have the root certificate of the certificate authority (CA) or if you have a self-signed certificate, import the CA certificate into the keystore and truststore.
   For example, if you have a CA certificate in a PKCS#10 file named `cust-ca.crt`, run this command from the <*`UnwiredPlatform_InstallDir`*>

`\UnwiredPlatform\Servers\UnwiredServer\Repository\Security` directory:

```
keytool -importcert -alias customerCA -file cust-ca.crt -storepass
changeit -keystore truststore.jks -trustcacerts
```

The truststore is used when Unwired Platform makes an out-bound connection over SSL to another server with a server certificate. Unwired Server checks that the server certificate is in the truststore, or is signed by a CA certificate in the truststore.

2. Generate a key pair in the Unwired Platform keystore.

   The command you use depends on the environment for which you are generating the keystore. For most Unwired Platform deployments, this command may be sufficient:

   ```
   keytool -genkeypair -alias supServer -keystore keystore.jks -
   keyalg RSA -keysize 2048
   -validity 365 -keypass mySecret -storepass changeit
   ```

   However, if you are generating a key pair to secure an HTTPS communication port between the SAP Gateway and Unwired Server for OData push notifications, you might use a command like:

   ```
   keytool –genkeypair –alias SAPpush –keyalg RSA –keysize 1024 –
   sigalg SHA1withRSA
   –keypass mySecret –keystore keystore.jks
   ```

3. Supply values for each of the resulting prompts.

   The first prompt is the most critical. If you are running multiple Unwired Server in a cluster, type an asterisk followed by the domain name where the Unwired Servers are running.

   ```
   What is your first and last name?
   [Unknown]: *.mydomain.com
   What is the name of your organizational unit?
   [Unknown]: myOU
   What is the name of your organization?
   [Unknown]: mycompany
   What is the name of your City or Locality?
   [Unknown]: place
   What is the name of your State or Province?
   [Unknown]: state
   What is the two-letter country code for this unit?
   [Unknown]: AB
   Is CN=*.mySUPdomain.com, OU=myOU, O=mycompany,
   L=place, ST=state, C=AB correct?
   [no]: y
   ```

   Note: The asterisk before the domain name allows this same certificate to be used by multiple Unwired Servers deployed as members of a common cluster. The CN value must be the domain name of the host on which Unwired Server is installed.

4. Generate a certificate signing request, send it to the certificate authority, and install the issued certificate in the Unwired Server keystore:

a) Generate a certificate signing request (CSR). For example:

```
keytool -certreq -alias supServer -keystore keystore.jks -
storepass changeit
-keypass mySecret -file supServer.csr
```

b) Send the CSR to the CA for signing.
For example, for SAP, may perform steps similar to:
1. Launch the URL for your SAP CA.
2. Change the option to **Certify the cert req** in the **select cmd** option.
3. Paste the content of the `.csr` file generated in the previous step.
4. Copy the content between (and including) `"-----BEGIN CERTIFICATE-----" "-----END CERTIFICATE-----"` of the response, to a text file named `<name of the cert>.cer`.
5. View and verify the status of the certificate.

c) Use `keytool` to import the CA.

Note: The -alias/-keypass values are the same as those used to generate the key pair and CSR. By sharing these values, you pair the signed certificate with the keypair:

```
keytool -importcert -alias supServer -file supServer.crt -
keypass mySecret -storepass changeit
-keystore keystore.jks -trustcacerts
Certificate reply was installed in keystore
```

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:

   If the security profile authenticates only the server, then only the server must provide a certificate to be accepted or rejected by the client. If the security profile authenticates both the client and the server, then 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.

Profile	Authenticates	Cipher suites
intl	server	• SA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA
intl_mutual	client/server	• RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA
strong	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
strong_mutual	client/server For example, this is the required option for mutual authentication of Unwired Platform and Gateway.	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
domestic	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • TLS_RSA_WITH_NULL_MD5 • TLS_RSA_WITH_NULL_SHA

Profile	Authenticates	Cipher suites
domestic_mutual	client/server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • RSA_WITH_NULL_MD5 • RSA_WITH_NULL_SHA

6. Use IIOPS in the Communication Ports subtab by selecting **Secure Management Port** (port 2001), and ensure that Sybase Control Center's Managed Resource properties match. By default, IIOPS is already configured between Unwired Server and Sybase Control Center. .
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.

See also

- *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners* on page 41
- *Securing Multiple Domains* on page 48

Securing Multiple Domains

To prevent role mapping leaks between multiple tenant domains, configure domains and assign shared security configurations.

Sybase recommends that the Platform administrator:

1. Create at least one new tenant domain in Sybase Control Center. You may require more, depending on your mobility strategy.
2. Restrict the use of the "admin" security configuration on the "default" domain to administration authentication only.
3. Assign at least one domain administrator. Depending on the maintenance issues of large-scale deployments, the administrator may want to use at least one Domain administrator per domain.

4. Create and assign at least one new security configuration. The administrator may create and assign security configurations, if security requirements (stringency, uniqueness) differ between tenant domains.

For more information, search for *Domains* in *Sybase Control Center* online help.

For example, a company named "Acme" has two separate divisions, HR and sales. The employees in each division use different mobile applications. In this case, Sybase recommends using two domains in Sybase Control Center to simplify the management of packages, users, applications and related artifacts.

Acme implements separate domain administrators for each domain, but is using a single "acme" security configuration due to the way the corporate LDAP directory is configured. This configuration includes an LDAPLoginModule provider that uses this URL:

```
ldap://ldap.acme.com
```

As a result, all employees of all domains are authenticated by the same LDAP server, and authorized by the same set of groups and roles.

Note: Because domain administrators are authenticated from the same acme LDAP repository via the admin security configuration on the default domain, those role mappings can "leak" between domains. Consequently, a domain administrator assigned to one domain gets granted access to another. This side-effect is undesirable and should be avoided.

Determining a Tenancy Strategy

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

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

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

1. Organize device users according to the data they need to access. Ideally, create a domain for each distinct set of users who access the same applications and authenticate against the same back-end security systems. If you do not need to support multiple groups in distinct partitions, then the single default domain should suffice.
2. Consider how these groups will be affected by administrative operations that prevent them from synchronizing data. Sometimes, you can limit the number of users affected by administration and maintenance disruptions by distributing packages across additional domains. Operationally, the more components a domain contains, the more clients who are

unable to access package data during administrative operations like domain synchronizations.

3. Assess the administrative resources of the tenant to determine how much time can be committed to domain administration tasks. Certain multitenant configurations require a greater amount of administrative time. For example, if you distribute packages from the same EIS across a number of domains, you must maintain identical data source configurations for each of these packages. This option requires more administrative time than grouping all packages belonging to the same EIS into one domain.
4. Decide how many domains to create for the customer, and identify which packages to group within each domain, according to the needs of the user groups you identified in step 1.

Benefits and Drawbacks of a Shared Security Configuration

Determine whether or not to use a shared security configuration across multiple domains.

Sybase recommends that you use differently named security configurations for each domain, unless you are willing to accept the risks, and domain administrators collaborate before implementing changes.

Benefit	Drawback
• Set up the modules you required in a named security configuration once.	• A domain administrator from one domain can make changes to role mapping at the default level, potentially with adverse effects to packages deployed to a different domain.

Creating and Enabling a New Domain

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

Prerequisites

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

Task

1. Open Sybase Control Center.
2. In the left navigation pane, select the **Domains** folder.
3. In the right administration pane, select the **General** tab, and click **New**.
4. In the Create Domain dialog, enter a name for the domain and click **Next**.

5. Optional. Select a security configuration for the domain by checking an option from the list of available configurations. These security configurations are then available for use in validating users accessing the packages.
6. Click **Next**.
7. Optional. Select one or more domain administrators for the domain.
8. Click **Finish**.
   The new domain appears in the **General** tab.
9. Click the box adjacent to the domain name, click **Enable**, then click **Yes** to confirm.

Enabling Authentication and RBAC for User Logins

Enable authentication and role-based access control (RBAC) for device user logins by creating a new security configuration (that is, one that is distinct from the "admin" security configuration on the "default" domain), and mapping roles, then assigning it.

Of all the default roles included with Unwired Platform, only the "SUP DCN User" role must be mapped, and only if DCNs are used for MBO packages associated with the security configuration you create for device user logins.

1. *Creating a Security Configuration for Device Users*

   Create and name a set of security providers and physical security roles to protect Unwired Platform resources. For device user authentication, create at least one provider that is not the "admin" security configuration on the "default" domain, which is used exclusively for administrator authentication in Sybase Control Center.
2. *Assigning Providers to a Security Configuration*

   Assign providers after you have created a security configuration.
3. *Stacking Providers and Combining Authentication Results*

   (Not applicable to Online Data Proxy) Optionally, implement multiple login modules to provide a security solution that meets complex security requirements. Sybase recommends provider stacking as a means of eliciting more precise results, especially for production environment that require different authentications schemes for administrators, DCN, SSO, and so on.
4. *Assigning Security Configurations to Domains, Packages, or Applications*

   A security configuration can be assigned to a domain. Domain administrators can then select the security configuration when deploying synchronization packages or creating application templates.
5. *Mapping Roles for Domains, Packages, or Applications*

   Role mappings can occur at two levels. Package level mappings override the mappings set at the default level. A default level is a level from which the role mapping is inherited.

See also

- *Securing Platform Administration* on page 32
- *Encrypting Synchronization for Replication Payloads* on page 102
- *Authentication in Unwired Platform* on page 53

Supported Providers and Credential Types

Different security providers allow users to supply different user credentials. If your security policy mandates that a specific credential type or strength be used, review the providers that are available to you.

Table 1. Credentials and Providers

Credential	Providers Available
User name and password	LDAP, NTProxy, HTTP
E-mail address and password. Note that E-mail addresses must follow certain requirements for Unwired Platform to recognize them correctly, especially when a security configuration is defined with it. See *Considerations for E-mail Addresses as Username*	LDAP, HTTP
X.509 certificates	Certificate, SAP SSO
Tokens	HTTP

Considerations for Using E-mail Addresses as User Names

At registration, application users can use an e-mail address as a user name in Unwired Platform. However, those users must ensure that e-mail addresses are processed correctly, especially when a security configuration is paired with the e-mail address.

A valid e-mail address:

- Can use any combination of uppercase and lowercase English alphanumeric characters (a–z, A–Z, and 0–9)
- Is limited to:
  - These special characters, which you can use without escape characters: .!#$%&'*+-/=?^_`{|}~ (that is, ASCII 33, 35–39, 42, 43, 45, 46, 47, 61, 63, 94–96, and 123–126)
  - These special characters, with which you must use an escape character: "(),:;<>@[\] (that is, ASCII 32, 34, 40, 41, 44, 58, 59, 60, 62, 64, and 91–93)
- For Unwired Platform 2.1 ESD #3, user name cannot exceed 100 characters for messaging applications or 128 characters for other application types.
- The user name length limit for packages deployed on earlier versions of Unwired Platform is still 36.

Note: This syntax information is only for your reference; while Unwired Server validates strings to ensure there are no restricted characters, it does not validate addresses to ensure they are syntactically correct.

When you use an e-mail address as the user name, ensure that the e-mail address domain is followed with a "." to prevent the address from being misinterpreted as a security configuration name. For example, jdoe@domain.com.

Table 2. E-mail Address Parsing Examples

User Name Entered	Result
userID	A user ID string. No risk of misinterpretation.
userID@textA	The user is authenticated with the "textA" security configuration if it exists.
userID@textb.com	An e-mail address as user name. No security configuration identified.
userID@textb.com@textc	An e-mail address. “textc” is treated as security configuration.
userID@textg.com@*texth*.com	An e-mail address. No security configuration identified, as the string after last @ contains a period.

Authentication in Unwired Platform

A security provider verifies the identities of application users and administrators who request access via one or more configured login modules.

Device user authentication and administrator authentication are configured differently:

- device users are authenticated with custom Unwired Server security configurations created by the platform administrator in Sybase Control Center. For SAP EIS backends, SSO authentication can be configured.
- Administrators are authenticated with the "admin" security configuration on the "default" domain. For first-time logins, administrators are authenticated with the PreconfiguredLoginModule. Once logged in, administrators for production systems should immediately reconfigure security to use the enterprise security backend and delete this login module from Sybase Control Center.

Caching Authenticated Sessions

An authentication request with username/password or certificate credentials for a specific domain always results in looking up an existing authenticated session in the cache that used the same credentials. If one is found, the session is reused instead of delegating the authentication request to the configured security backend. This is the case even if any of the information from the client session is used to authenticate the user instead of the presented username/password or certificate credentials.

If an existing authenticated session is found in the cache with the same credentials, then the user is not authenticated again against the configured security backend even if the cached session was authenticated based on an http header/cookie/personalization value and the new authentication request contains a different value for that parameter.

See also

- *Enabling Authentication and RBAC for User Logins* on page 51
- *Enabling Authentication and RBAC for Administrator Logins* on page 33

Creating a Security Configuration for Device Users

Create and name a set of security providers and physical security roles to protect Unwired Platform resources. For device user authentication, create at least one provider that is not the "admin" security configuration on the "default" domain, which is used exclusively for administrator authentication in Sybase Control Center.

Only platform administrators can create security configurations. Domain administrators can view configurations only after the platform administrator creates and assigns them to a domain.

1. In the left navigation pane, expand the **Security** folder.
2. In the right administration pane, click **New**.
3. Enter a name for the security configuration and click **OK**.

See also

- *Assigning Providers to a Security Configuration* on page 61

Built-in Security Providers for User Authentication and Authorization

Unwired Server supports a variety of built-in security providers use to authenticate device users. Administrators define one or more security providers when they create security configurations using Sybase Control Center.

You can configure a provider of a given type only if that provider is available on the enterprise network.

If you are using Unwired Platform in an Online Data Proxy deployment, not all providers are applicable in this environment.

No Security Provider

A NoSec provider offers pass-through security for Unwired Server, and is intended for use in development environments or for deployments that require no security control. Do not use this provider in production environments— either for administration, or device user authentication.

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 provides 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

(Not applicable to Online Data Proxy) The LDAP security provider includes authentication, attribution, and authorization providers. Add an LDAP provider to a security configuration to authenticate administrator logins (on the "admin' security configuration on the "default" domain) or device user logins (any custom security configuration for that purpose).

You can configure these providers:

- **LDAPLoginModule**– provides authentication services. Through appropriate configuration, you can enable certificate authentication in **LDAPLoginModule**.
- (Optional)**LDAPAuthorizer** or **RoleCheckAuthorizer** – provide authorization services for **LDAPLoginModule**. **LDAPLoginModule** works with either authorizer. In most production deployments, you must always 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**.

  The RoleCheckAuthorizer is used with every security configuration but does not appear in Sybase Control Center.

  Use **LDAPAuthorizer** only when **LDAPLoginModule** is not used to perform authentication, but roles are still required to perform authorization checks against the LDAP data store. If you use **LDAPAuthorizer**, always explicitly configure properties; for it cannot share the configuration options specified for the **LDAPLoginModule**.

You need not enable all LDAP providers. You can also implement some LDAP providers with providers of other types. If you are stacking multiple LDAP providers, be aware of, and understand the configuration implications.

See also

- *LDAP Configuration Properties* on page 151
- *Adding a Production-Grade Provider* on page 35
- *Stacking Providers and Combining Authentication Results* on page 98

Configuration Best Practices for Multiple LDAP Trees

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

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 AuthenticationScopes 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 AuthenticationScope within the LDAP tree structure, this approach can have performance implications.
- If multiple servers are clustered together to form a large logical directory tree, configure the LDAPLoginModule by setting the `Referral` property to `follow`.
- If subjects have been made members of too many LDAP groups and the search for physical roles results in too many results, the maximum result limit may be reached and authentication fails. To avoid this, narrow the `RoleSearchBase` to LDAP groups that are relevant only to Unwired Platform.

Configuring LDAP to use SSL

If your LDAP server uses a secure connection, and its SSL certificate is signed by a nonstandard certificate authority, for example it is self-signed, use the keytool utility (**keytool.exe**) to import the certificate into the truststore.

1. Run the following console command: **keytool.exe -import -keystore <UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Security \truststore.jks -file <your cert file and path> -alias ldapcert -storepass changeit**.
2. Restart Sybase Unwired Platform services.
3. Log in to Sybase Control Center for Sybase Unwired Platform.
4. In the navigation pane of Sybase Control Center, expand the Security folder and select **admin**.
5. In the administration pane, click the **Authentication** tab.
6. Add an LDAPLoginModule, configuring the ProviderURL, Security Protocol, ServerType, Bind DN, Bind Password, Search Base, and other properties determined by you and the LDAP administrator. See *LDAP Configuration Properties* and *LDAP Login and Authorization Modules*.
   a) Use `ldaps://` instead of `ldap://` in the **ProviderURL**.
   b) Use `ssl` in the **Security Protocol**.
7. In the **General** tab, select **Validate** then **Apply**.
8. Click **OK**.

LDAP Role Computation

Role checks are the primary means of performing access control when using LDAP authentication. Authentication and attribution capabilities both utilize role computation techniques to enumerate roles that both authenticated users have.

There are three distinct types of role constructs supported by LDAP providers; each may be used independently, or all three may be configured to be used at the same time.

- User-level role attributes, specified by the UserRoleMembershipAttributes configuration property, are the most efficient role definition format. A user's roles are enumerated by a read-only directory server-managed attribute on the user's LDAP record. The advantage to this technique is the efficiency with which role memberships can be queried, and the ease of management using the native LDAP server's management tools. These constructs are supported directly by ActiveDirectory products, and use these configuration options:
  - UserRoleMembershipAttributes – the multivalued attribute on the user's LDAP record that lists the role DNs that the user is a member of. An example value for this property is "memberOf" on ActiveDirectory.
  - RoleSearchBase – the search base under which all user roles are found, for example, "ou=Roles,dc=sybase,dc=com". This value may also be the root search base of the directory server.
  - RoleFilter – the search filter that, coupled with the search base, retrieves all roles on the server.
  - (Optional) RoleScope – enables role retrieval from subcontexts under the search base.
  - (Optional) RoleNameAttribute –choose an attribute other than "cn" to define the name of roles.

  These properties retrieve correct values automatically based on the type of server you configure.
- LDAP servers allow groups to be members of other groups, including nested groups. The LDAP provider does not compute the group membership information recursively. Instead, nested group membership information is taken into consideration for role computation only if the LDAP server provides a user attribute that contains the complete list of group memberships, including static, dynamic, and nested group memberships. See *Using LDAP Nested Groups and Roles*.
- Freeform role definitions are unique in that the role itself does not have an actual entry in the LDAP database. A freeform role starts with the definition of one or more user-level attributes. When roles are calculated for a user, the collective values of the attributes (each of which may can be multivalued) are added as roles to which the user belongs. This technique is particularly useful when the overhead of managing roles uses are administration-heavy. For example, assign a freeform role definition that is equivalent to the department number of the user. A role check performed on a specific department number is satisfied only by users who have the appropriate department number attribute value. The only property that is required or used for this role mapping technique is the comma-delimited UserFreeformRoleMembershipAttributes property.

Using LDAP Nested Groups and Roles

LDAP group definitions may be used as role definitions within nested groups across LDAP servers.

LDAP servers allow groups to be members of other groups, including nested groups. The LDAP provider does not compute the group membership information recursively. Instead, nested group membership information is taken into consideration for role computation only if the LDAP server provides a user attribute that contains the complete list of group memberships, including static, dynamic, and nested group memberships. Unwired Platform implements a Java LDAP provider through a common security interface. LDAP group memberships are stored and checked on a group-by-group basis. Each defined group, typically of objectclass "groupofnames" or "groupofuniquenames," has an attribute listing all of the members of the group. Unwired Platform retrieves static group membership from the user attribute UserRoleMembershipAttributes.

For Active Directory server, the UserRoleMembershipAttributes property for the LDAP provider should be set to "tokenGroups" to enable it to retrieve the nested group membership information. For SunOne server, the UserRoleMembershipAttributes property for the LDAP provider should be set to "nsRole" instead of the default value "nsRoleDN" to enable it to retrieve the nested roles information. For additional information, see *Skipping LDAP Role Lookups (SkipRoleLookup)*, and *LDAP Configuration Properties*.

Note: Unwired Platform retrieves static group membership from the user attribute UserRoleMembershipAttributes.

Note: Unwired Platform supports static group and role membership lookups only. It does not support dynamic group membership lookups.

Skipping LDAP Role Lookups (SkipRoleLookup)

When configuring LDAP, use the **SkipRoleLookup** configuration option to grant the user attributes defined in the **UserRoleMembershipAttributes** property.

Use the configuration option **SkipRoleLookup** for all LDAP providers to grant the user roles specified in the **UserRoleMembershipAttributes** property.

Set the **SkipRoleLookup** configuration option to `true` to grant the user the roles defined in the **UserRoleMembershipAttributes** property. When set to true, the user roles are not cross-referenced with the roles retrieved in the role search base and role search filter.

Note: Setting **SkipRoleLookup** to `true` grants **all** the roles retrieved using the UserRoleMembershipAttributes property from the LDAP user entry. This eliminates the need to look up all the roles defined in the role search base and match the role filter as roles are retrieved.

LDAP Login and Authorization Modules

LDAP login and authorization modules can sometimes share a common configuration. However, authorizers do not inherit configuration from login modules you configure. Configurations must be explicit.

In the case where both LDAPLoginModule and LDAPAuthorizer are configured:

- Matching configuration – LDAPAuthorizer simply skips the role retrieval.
- Differing configuration – LDAPAuthorizer proceeds with the role retrieval from the configured backend, and performs the authorization checks using the complete list of roles (from both the login module and itself). Even in the case where multiple LDAPLoginModules are configured, only one LDAPAuthorizer is required as it compares its configuration with the configuration used for the successful authentication of the user.

NTProxy Security Provider

(Not applicable to Online Data Proxy) NTProxy — sometimes known as native Windows login — is an Unwired Server provider that integrates with existing Windows login security mechanisms. Add an LDAP provider to a security configuration to authenticate administrator logins (on the "admin" security configuration on the "default" domain) or device user logins (any custom security configuration for that purpose).

If added to a particular security configuration, users or administrators 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.

SAP SSO Token Security Provider

The SAPSSOTokenLoginModule has been deprecated and will be removed in a future release. Use HttpAuthenticationLoginModule for SAP SSO2 token authentication.

Use HttpAuthenticationLoginModule for both JCo and DOE-C connections to the SAP system. Unwired Server does not provide authorization control or role mappings for user authorization; enforce any access control policies in the SAP system.

See also

- *SAP SSO Token Authentication Properties* on page 168
- *Certificate Authentication Properties* on page 162
- *HTTP Basic Authentication Properties* on page 172

Certificate Security Provider

Use the Unwired Server CertificateAuthenticationLoginModule authentication provider to implement SSO with an SAP enterprise information system (EIS) with X.509 certificates.

Unwired Server does not provide authorization control or role mappings for user authorization; enforce any access control policies in the EIS.

See also

- *SAP SSO Token Authentication Properties* on page 168
- *Certificate Authentication Properties* on page 162
- *HTTP Basic Authentication Properties* on page 172

HTTP Authentication Security Provider

Use HttpAuthenticationLoginModule provider to use Basic authentication to enable automatic application registration. This provider is required when registration is set to automatic. It can also be used to enable SSO into SAP servers in place of the deprecated SAPSSOTokenLoginModule.

The LoginModule validates standard username/password style credentials by passing them to a Web server. Configure the URL property to point to a Web server that challenges for basic authentication.

This provider is enhanced to authenticate the user by validating a token specified by the client by sending the configured client values to the HTTP backend in the specified format (header/cookie). Any parameter value, for example personalization parameter, http header, or http cookie can be specified in the ClientHttpValuesToSend property so that the provider can retrieve the value of the configured parameter(s) and pass them to the Web server in the format required by the SendClientHttpValuesAs configuration property.

For example, to extract the cookie "MyCookie" from the client session to Unwired Server and pass it along to the Web server as the cookie "testSSOCookie", set the properties ClientHttpValuesToSend to "MyCookie" and set SendClientHttpValuesAs to cookie:testSSOCookie.

Note: Note that if "ClientHttpValuesToSend" property is configured, the provider only attempts to authenticate the user using those values. It does not set the username/password credentials in the http session to the Web server. If the specified client values are not found in the client session to SUP or if the Web server fails to validate the specified token, then this provider fails the authentication unless the property "TryBasicAuthIf TokenAuthFails" is set to `true` to enable it to revert to passing the username/password credentials to respond to the BasicAuth challenge.

Best practice guidelines include:

- Using an HTTPS URL to avoid exposing credentials.

- If the Web server's certificate is not signed by a well known CA, import the CA certificate used to sign the Web server's certificate into the Unwired Server `truststore.jks`. The truststore is prepopulated with CA certificates from reputable CAs.
- If this Web server returns a cookie as part of successful authentication, set the `SSO Cookie Name` configuration property to the name of this cookie. Upon successful authentication, this login module places the cookie value into an `HttpSSOTokenCredential` object and attaches it to the `java.security.Subject` as a public credential.

  Note: The HTTP Basic login module is the module that can either be used for SSO tokens or HTTP basic without SSO. The sole condition being that the backend support HTTP Basic authentication.
- When using this module in lieu of the deprecated SAPSSOTokenLoginModule, the cookie name is typically "MYSAPSSO2".

For example, SiteMinder is often used in mobile deployments to protect existing Web-based applications. Existing users point their browser at a URL, and SiteMinder intercepts an unauthenticated session to challenge for credentials (Basic). When the authentication succeeds, it returns a SMSESSION cookie with a Base64-encoded value that can be used for SSO into other SiteMinder enabled systems.

See *HTTP Basic Authentication Properties*.

See also

- *SAP SSO Token Authentication Properties* on page 168
- *Certificate Authentication Properties* on page 162
- *HTTP Basic Authentication Properties* on page 172
- *Enabling Authorization for Data Change Notification CDB Insertions* on page 142
- *Assigning Providers to a Security Configuration* on page 61
- *Stacking Providers and Combining Authentication Results* on page 98

Assigning Providers to a Security Configuration

Assign providers after you have created a security configuration.

1. In the left navigation pane, expand the **Security** folder.
2. Select the security configuration you want to assign a provider to.
3. In the right administration pane, select the **Settings** tab to set an authentication cache timeout value.

   The timeout determines how long authentication results should be cached before a user is required to reauthenticate. For details, see *Authentication Cache Timeouts*. To configure this value:

   a) Set the cache timeout value in seconds. The default is 3600.
   b) Click **Save**.

4. Select the tab corresponding to the type of security provider you want to configure: Authentication, Authorization, Attribution, or Audit.
5. To edit the properties of a preexisting security provider in the configuration:
   a) Select the provider, and click **Properties**.
   b) Configure the properties associated with the provider by setting values according to your security requirements. Add properties as documented in the individual reference topics for each provider.
   c) Click **Save**.
6. To add a new security provider to the configuration:
   a) Click **New**.
   b) Select the provider you want to add.
   c) Configure the properties associated with the provider by setting values according to your security requirements. Add properties as documented in the individual reference topics for each provider.
   d) Click **OK**.
      The configuration is saved locally, but not yet committed to the server.
7. Select the **General** tab, and click **Validate** to confirm that Unwired Server accepts the new security configuration.
8. Click **Apply** to save changes to the security configuration, and apply them across Unwired Server.

Next

If you have multiple providers, understand how to stack/sequence them and know what the implication of provider order means.

See also

- *Creating a Security Configuration for Device Users* on page 54
- *Stacking Providers and Combining Authentication Results* on page 98

Assigning Security Configurations to Domains, Packages, or Applications

A security configuration can be assigned to a domain. Domain administrators can then select the security configuration when deploying synchronization packages or creating application templates.

By selecting a security configuration at the package or application connection template level, you can choose the granularity required for user authentication. For details on how to assign and select a security configuration at the domain, package, or application level, search for "Security Configuration" in the *Sybase Control Center* online help.

See also

- *Stacking Providers and Combining Authentication Results* on page 98

Mapping Roles for Domains, Packages, or Applications

Role mappings can occur at two levels. Package level mappings override the mappings set at the default level. A default level is a level from which the role mapping is inherited.

Unwired Platform uses a role mapper to map logical and physical roles during an access control check. This allows developers to create applications that incorporate a logical access control policy. When the application is deployed, a security administrator can work with the developer to understand what the logical roles in the application were intended to do and map these logical roles to physical roles that exist in the real security system.

Default SUP Roles

Default Unwired Platform roles are logical roles that are built into the system. While five roles are available by default for the 'admin' security configuration only three roles are necessary for enabling role-based access to administrative and DCN interfaces.

Default roles include:

- **SUP Administrator –** The platform administrator for the runtime and mobile artifacts. A required role for the "admin" security configuration. See *SUP Administrator and SUP Domain Administrator*.
- **SUP Domain Administrator –** The administrator assigned to a domain. A required role for the "admin" security configuration. See *SUP Administrator and SUP Domain Administrator*.
- **SUP DCN User –** The administrative user for DCNs. A required role for the "admin" security configuration. This role must be mapped in *every* security configuration assigned to deployed MBO packages that receive DCN requests. See *SUP DCN User Role*.
- **SUP User, SUP Developer –** Sample logical roles that have no impact on administration. Do not map these roles or you may expose administration privileges inadvertently to non-administration users.

All required administrative roles must be mapped for the "admin" security configuration on the default domain, otherwise administrator authentication fails:

- In a development environment, mappings are automatically created.
- In a production environment, physical roles must be added manually to the directory used, and then manually mapped in Sybase Control Center.

All optional maps may be mapped for the "admin" security configuration. However, they may be required for custom configurations you create to authenticate non-administrative users.

Note: Do not use the "admin" security configuration for the authentication of development or application users. Also do not assign the admin security configuration to any domain other than "default". The default domain is reserved for administrative use only.

SUP DCN User Role

The SUP DCN User is a logical role that Unwired Platform uses to authorize any DCN event: updating data in the cache, executing an operation, or triggering a workflow package.

Before any DCN event is submitted, the person or group mapped to this role must be authenticated and authorized by the security configuration. By default, SUP DCN User is automatically available to all new security configurations. However, the underlying default varies depending on the environment in use.

Before this logical role can be used, SUP DCN User must be mapped to a physical role in the enterprise security repository, and the user who performs DCN must be in that physical role.

- To map the SUP DCN User to a user in the underlying security repository, the user name must be first defined in Sybase Control Center as a physical role that is mappable. Then, SUP DCN User role can be mapped to a physical user or to a physical role from Sybase Control Center. For example, to map SUP DCN User to a user that is not in the security repository, use the format `user:`*`User`*.

  If you are supporting multiple domains, the user name must also include the named security configuration for the package the DCN is targeted for, by appending *@DomainSecurityConfigName* as a suffix to that name. Suppose you have two packages (PKG_A, PKG_B) deployed to two domains (Domain_A, Domain_B) respectively. Further, assume that PKG_A in Domain_A has been assigned to the "admin" security configuration, whereas PKG_B in Domain_B has been assigned to the "alternateSecurityConfig" security configuration.
  - A user doing DCN to PKG_A should identify him or herself as *User@admin*.
  - A user doing DCN to PKG_B should identify him or herself as *User@alternateSecurityConfig*.

  If you are using ActiveDirectory, and are using e-mail addresses for user names, definitions appear as *username@myaddress@DomainSecurityConfigName*.

The implementation varies, depending on the DCN service used:

- For workflows, because the resource the user is pushing data toward is a group of named users (users authenticated previously successfully against a certain security configuration), he or she must have the authorization to push to that particular security configuration. The user must be mapped to SUP DCN User in the security configuration for the workflow target.
- A user having SUP DCN User logical role in security configuration "mySecConfig1"' must not have the right to push workflow DCN or regular DCN to a user or package associated with "mySecConfig2".

See also

- *SUP Administrator and SUP Domain Administrator* on page 65

SUP Administrator and SUP Domain Administrator

The SUP Administrator logical roleis a superuser of Sybase Control Center and can therefore perform all administrative operations in the Unwired Platform administration console. The SUP Domain Administrator logical role can perform only those Unwired Platform administrative operations that pertain to the domain it is assigned to.

Note: The terms "Unwired Platform (platform) administrator" and "domain administrator" are used in all documentation to refer to the user with "SUP Administrator" role and "SUP Domain Administrator" role, respectively.

The SUP Administrator and SUP Domain Administrator logical roles are mapped to specific physical roles in the security repository. These roles may also protect mobile business objects.

The domain-level role mappings for these logical roles in the "admin" security configuration in the "default" domain enable the platform administrator or domain administrator access control to Unwired Platform. Both types of administrators are authenticated and authorized by the security provider of the 'admin' security configuration. All administrative and application users and their passwords are managed in the enterprise security repository. Unwired Platform delegates security checks by passing login and password information to the security provider of the "admin" security configuration.

Note: Always create a new security configuration separate from "admin", and always create a different domain besides "default". This action separates the authentication of administration users from device users to unique domains, which is a practice that Sybase strongly recommends.

Sybase Control Center limits feature visibility depending on the role an administrator logs in with. The platform administrator login has access to the full cluster, whereas the domain administrator login can view information pertaining to only the assigned domains.

Only "supAdmin" is the default login for cluster-wide administration. This login is initially assigned both "SUP Administrator" and "SUP Domain Administrator" roles.

See also

- *SUP DCN User Role* on page 64
- *Enabling Authentication and RBAC for Administrator Logins* on page 33

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. The logical role and the physical role must match, otherwise, authorization fails.

State	Description
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). Carefully consider potential consequences before using this option.
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 details, see *Configuring Domain Security* in *Sybase Control Center* online help.
- For package-level mappings, configure role mappings when you deploy a package to Unwired Server, or at the package-level after deployment. For details, see *Assigning Package-Level Security* in *Sybase Control Center* online help.

Single Sign-on Integration Across Client Applications

Administrators can use their single sign-on system (SSO) of choice with Unwired Platform to achieve end-to-end integration across client applications and Enterprise Information Systems (EIS) resources.

In addition to supporting X.509 certificate security, Unwired Platform expands single sign-on support to third-party and standard single sign-on mechanisms. With expanded single sign-on support, Unwired Platform enables the authentication framework to accept HTTP headers and cookies propagated by the client or a proxy server and then authenticate and propagate the user to the EIS.

Network Edge Single Sign-on (SSO) Authentication

Unwired Platform applications can integrate with HTTP-based SSO authentication providers.

Mobile Workflow and Object API applications can connect to reverse proxy servers (agents) at the network edge. These agents perform authentication and return authenticated tokens on behalf of those authentication providers to either Unwired Server or HTTP-based enterprise information system (EIS) systems via session personalization values delivered as HTTP cookies, or HTTP headers. An example of an HTTP-based SSO provider is SiteMinder running inside the enterprise and its SiteMinder agent running at the network edge inside an Apache or IIS reverse proxy server.

For more information, see *Single Sign-on* in *Developer's Guide: Mobile Workflow Packages*.

Single Sign-on Using NamedCredential

In expanded single sign-on support, Unwired Platform allows the tokens generated by any system to be used for single sign-on. Administrators can configure the Web service connection properties with the name of the credential containing the token and how to propagate it to the Web service.

Any login module can add a NamedCredential to the authenticated subject. A NamedCredential is a credential that has a name associated with it and can contain any value. Typically, a credential is used to store a value that can be used to authenticate the user to a backend server using SSO.

For example, the SAPSSOTokenLoginModule by default adds the MYSAPSSO2 token as the credential with the name "MYSAPSSO2". The HttpAuthenticationLoginModule by default adds the cookie, when configured to look for one, as the single sign-on credential with the name set to the cookie name upon successfully authenticating the user.

To use the NamedCredential added by a login module for single sign-on into EIS, the administrator must set the properties in the EIS connection definition to identify the NamedCredential and how it should be propagated to the EIS in the following format:

```
credential.<X>.name=credential name
```

```
credential.<X>.mapping=credential mapping to header/cookie
```

where X is any unique ID that binds the name and the mapping for a specific credential. Multiple such bindings can be configured so that any or all of the available credentials can be passed to the backend using the specified mechanism.

SiteMinderSSOTokenCredential Example

The following is an example for specifying a sample SiteMinder token from the credential named SiteMinderSSOTokenCredential that should be set in the connection to the backend server as a SMSESSIONID cookie.

```
credential.1.name=SiteMinderSSOTokenCredential
```

```
credential.1.mapping=cookie:SMSESSIONID
```

SAPSSOTokenCredential Example

The following is an example to set the MYSAPSSO2 http header in the connection to the backend server using the value from the SAPSSOTokenCredential.

```
credential.myindex.name=SAPSSOTokenCredential
```

```
credential.myindex.mapping=header:MYSAPSSO2
```

Propagate Single Sign-on Using ClientValuePropagatingLoginModule

Applications can use session personalization values or HTTP headers and cookies to pass data that should be used for single sign-on into the Enterprise Information System (EIS) backend. The ClientValuePropagatingLoginModule enables administrators to add client values as named credentials, name principals, and role principals to the authenticated subject.

Adding client values as named credentials allows them to be used for single sign-on. When authenticating the user using a token from the client session, if the corresponding login module is unable to retrieve the user name from the token and add it as a principal for use in impersonation checking, the administrator can configure this provider to add the appropriate header value from the client session as a principal to the authenticated subject.

If there are session personalization values that an application is using as single sign-on data, the values are available to the Web server by using:

- The LoginModule to copy the personalization values or HTTP cookie and header from the client request and attach it to the authenticated subject as a named credential.
- Properties on the connection definition to specify the named credential found on the subject and how to pass it to the Web server.

Note: To avoid a client setting the client personalization key or HTTP header/cookie value to workaround the impersonation check, only use this configuration when the SSO framework requires it and the deployed applications ensure that the client cannot manipulate the headers set into the session. HTTP headers set by the network edge take precedence over the client personalization key. For more information, see *Impersonation Prevention Using the checkImpersonation Property*.

Note: This login module does not authenticate the subject but adds the NamedCredential if the user is successfully authenticated by other login modules. It always returns "false" from the login method and should always be configured with the controlFlag set to "optional" to avoid affecting the outcome of authentication process. See *controlFlag Attribute Values*.

Table 3. Configuration Options for ClientValuePropagatingLoginModule

Configuration Option	Default Value	Description
ClientHttpValuesAsNamed-Credentials	None	Comma separated list of mappings that specify the names of the client values and the name of the credential to add them. For example: `httpHeaderName:credentialName1` `httpCookieName:credentialName2` `personalizationParameterName1:credentialName3`
ClientHttpValuesAsNamePrin-cipals	None	Comma separated list of values from the client HTTP map that should be added as name principals after successful authentication.
ClientHttpValuesAsRolePrin-cipals	None	Comma separated list of values from the client HTTP map that should be added as role principals after successful authentication.

Impersonation Prevention Using the checkImpersonation Property

Administrators can set the **checkImpersonation** property associated with the security configuration to “false” to allow authentication to succeed when in token based authentication the user name presented cannot be matched against any of the user names validated in the login modules.

The **checkImpersonation** property is used when a custom login module that maps the token to a user name and adds a principal with that user name is unavailable. In token-based authentication, even though a valid token may be presented to Unwired Platform, the token may not be associated with the user indicated by the user name. To prevent the user authentication from succeeding, the checkImpersonation property is set to true by default.

. When an un-authenticated request is received by Unwired Platform (from a device or DCN request), it may contain a token (in an HTTP header or cookie) that should be validated to authenticate the user. In some cases a user name can be extracted from the token. In Unwired Platform, the specified user name is matched to the name of at least one of the public Principals added by the login modules. If the user name cannot be extracted from the token as part of the validation, then the specified user name is not added as a principal.

In certain situations, it may not be possible for the token validation server to return the user name embedded in the token. If no such custom login module is available, then the administrator can allow authentication to succeed even when the user name presented cannot be matched against any of the user names validated by the configured login modules. In these situations, a custom login module that maps the token to a user name and adds a principal with that user name may be used. To allow this authentication, **set the checkImpersonation property** associated with the security configuration to **false**.

Single Sign-on for SAP

Unwired Platform supports single sign-on (SSO) authentication for mobile clients that access data from an SAP enterprise information system (EIS) using either X.509 certificates or SSO logon tickets (SSO2).

Single sign-on credential support for SAP includes:

- X.509 certificates – use the CertificateAuthenticationLoginModule provider to implement X.509 authentication. At runtime, the mobile client selects the certificate signed by a trusted CA, which is authenticated by the SAP EIS.
- SAP single sign-on (SSO2) tokens – use the HttpAuthenticationLoginModule provider for both basic HTTP authentication and to implement SSO2. At runtime, the client enters a user name/password combination that maps to a user name/password in the SAP EIS. For SSO2, a token is obtained from the configured SAP server using the client-supplied user name/password and is forwarded to other SAP servers configured in the endpoints to authenticate the client, instead of using client-supplied user name/password credentials.

Single Sign-on Authentication

Understand the role of user credentials and X.509 certificates in single sign-on authentication.

Encrypt the communication channel between Unwired Server and the SAP EIS for security reasons:

- For Web services, DOE, and Gateway interactions this requires an HTTPS communication path with mutual certificate authentication. Use Sybase Control Center to navigate to the corresponding connection pool, edit the properties and add the properties "Certificate Alias" (give the name of a certificate alias in the keystore.jks), and "Password" (provide the key password).

During mutual certificate authentication, the client presents a certificate to Unwired Server. In order for authentication to succeed, the client’s certificate, or more typically the CA certificate that signed the client certificate must be present in the Unwired Server truststore. The Unwired Server truststore also contains a server-certificate (CN=host.domain) which is issued by the server (SAP for example), and which other SAP servers are configured to trust, meaning that once the server-certificate is authenticated during the HTTPS mutual certificate authentication, the SAP server further trusts that the credentials (SSO2 or X.509 values) given to identify the end user are correct, and the SAP server executes its EIS operations as that asserted end-user.

There is a separate notion of a "technical user" (CN=someTechUserName), which is different than the (CN=host.domain) server-certificate used for SSO. In a "normal" pooled JCo connection, the user name is a technical user, and all RFCs are executed in the SAP EIS as that user. The technical user is granted all rights and roles within SAP to allow it to execute the range of RFCs behind the MBOs, which is the opposite of SSO.

See also

- *Enabling Single Sign-on for DOE-C Packages* on page 23
- *SAP Single Sign-on and DOE-C Package Overview* on page 75
- *SAP Single Sign-on and Online Data Proxy Overview* on page 89
- *Enabling Single Sign-on for OData Applications* on page 24
- *SAP Single Sign-on and Mobile Business Object Package Overview* on page 77
- *Enabling Single Sign-on for Mobile Business Object Packages* on page 25

SAP External Libraries Overview

Understand the purpose of the external files you can optionally download from SAP and install into Unwired Platform to enable communication with an SAP EIS.

- **SAP Cryptographic Libraries –** required by Unwired Platform to enable Secure Network Communications (SNC) between Unwired Server or Unwired WorkSpace and the SAP EIS.
- **SAPCAR utility –** required to extract files from the SAP cryptographic library.

Installing the SAP Cryptographic Libraries on Unwired Platform

Installation and configuration is required if you want to configure Secure Network Communications (SNC) for Unwired Platform SAP JCo connections. SNC may be required by the SAP EIS in question, if SSO2 tokens or X.509 certificates are used for connection authentication.

Prerequisites

Download and install the SAPCAR utility, which is required to extract the contents of the cryptographic library.

Task

Unzip and install the contents of the latest SAP Cryptographic archive on your Unwired Server host. There are different distribution packages for various hardware processors.

Make sure you are installing the correct libraries for your environment, and into folders based on the particular architecture of your machine.

1. Go to the SAP Web site at *http://service.sap.com/swdc* and download the latest SAP cryptographic library suitable for your platform.

   a) Navigate to **Installations and Upgrades > Browse our Download Catalog > SAP Cryptographic Software > SAP Cryptographic Software**.
   b) Select and download the platform specific file.
2. Create a directory in which you unzip the Cryptographic zip file. For example: `C:\sapcryptolib`.
3. Copy the appropriate Windows cryptographic library for your machine (for example, 90000101.SAR) to the `C:\sapcryptolib` directory.
4. Open a command prompt and navigate to `C:\sapcryptolib`.
5. Extract the SAR file. For example:

   SAPCAR_4-20002092.EXE -xvf C:\90000101.SAR -R C:\sapcryptolib
6. Create a new directory somewhere on your computer (for example, `C:\sapcryptolib`), then:
   - For Itanium 64 bit processors, copy the `ntia64` subdirectory contents.
   - For Intel 64 bit processors, copy the `nt-x86_64` subdirectory contents.
   - For Intel 32 bit processors, copy the `ntintel` subdirectory contents.
7. Delete the corresponding subdirectory when files have been moved.
8. (Optional) Add the SECUDIR environment variable to the user environment batch file: `<`*`UnwiredPlatform_InstallDir`*`>\UnwiredPlatform\Servers\UnwiredServer\bin\usersetenv.bat`.
9. If you have installed Unwired WorkSpace, you must add the SECUDIR variable to the WorkSpace batch file: `<`*`UnwiredPlatform_InstallDir`*`>\UnwiredPlatform\Eclipse\UnwiredWorkspace.bat`.

Installing the SAPCAR Utility

Unzip and install the latest SAPCAR utility on your Unwired Server or Unwired WorkSpace host, which you can use to extract the contents of compressed SAP files, for example RFC and cryptographic library files.

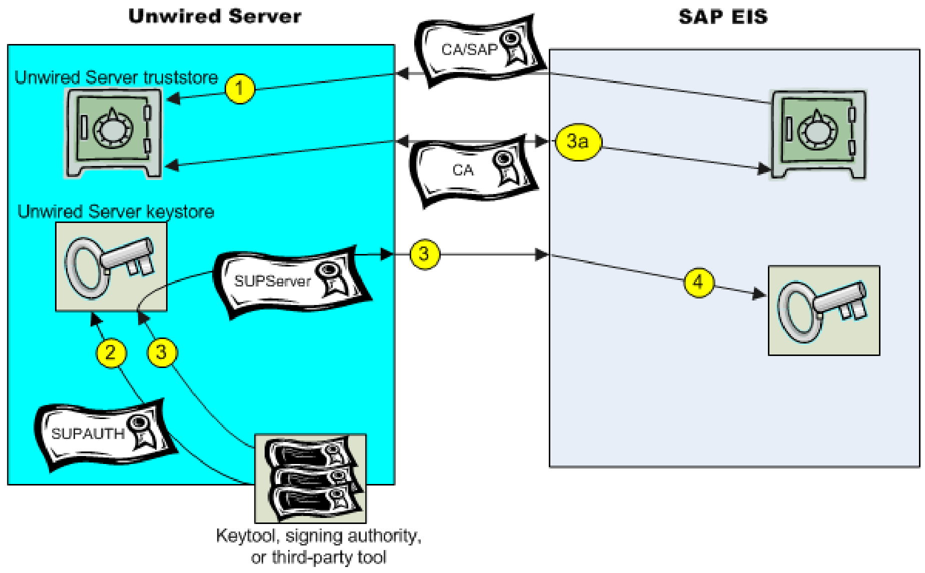
The installation package is available to authorized customers on the SAP Service Marketplace. There are different distribution packages for various hardware processors. Select the package appropriate for your platform.

1. Go to the SAP Web site at *http://service.sap.com/swdc*.
2. From the SAP Download Center, navigate to **Support Packages and Patches > Browse our Download Catalog > Additional Components**
3. Select **SAPCAR**.
4. Select the current version. For example, **SAPCAR 7.10**, then select and download the SAPCAR appropriate for your platform.

Configuring X.509 Certificates for SAP Single Sign-on

Import, export, and generate the X.509 certificates that secure communication paths between Unwired Server and the SAP enterprise information system (EIS), and for client authentication, including single sign-on (SSO) with X.509 or SSO2 tokens.

Figure 1: Creating, Importing, and Exporting Certificates



Use Java **keytool** commands to import these certificates into the Unwired Server truststore and keystore.

1. Import SAP CA certificates into the Unwired Server truststore, including:
- The standard SAP/DOE server root certificate (.crt or .cer) required to establish a trusted relationship between Unwired Server and the SAP EIS.
- Any CA certificate used to sign .pse certificates used for JCo/SNC communications.
- For Gateway deployments where Unwired Server is the Online Data Proxy (ODP), import the Gateway server's CA into the truststore of Unwired Platform.

  The ODP requires two certificate files: one that contains the certificate and private key for use by the server, and another that contains only the certificate for use by clients. The certificates should be in the form of a PKCS#10 file using an RSA key pair (key lengths in the range of 512–16384 are supported), in PEM or DER format. The key usage should be set to Key Encipherment, Data Encipherment, Key Agreement (38).
- Any other required SAP CA certificate. For example, any CA certificate used to sign a client certificate that is to be authenticated by Unwired Server must be imported if you are implementing SSO with X.509.

Note: If Unwired Server is communicating with a server that is hosting a Web service that is bound to SAP function modules, import that server's CA certificate into the Unwired Server truststore.

For example:

```
keytool -import -keystore <UnwiredPlatform_InstallDir>/
UnwiredPlatform/Servers/UnwiredServer/Repository/Security/
truststore.jks -file <CertificateFile>
```

```

Enter keystore password:  changeit
Trust this certificate? [no]:  yes
```

2. Create a keystore on the Unwired Server host into which you can import the certificate and private key (PKCS #12) issued by the SAP system administrator, then import the certificate into the Unwired Server keystore. This certificate secures communications for packages and is used when a user uses an X.509 certificate rather than a user name and password. For example:

   ```
   keytool -importkeystore -srckeystore SUPAUTH.p12 -
   srcstoretype pkcs12 -srcstorepass <techuserpass> -srcalias
   CERTALIAS -destkeystore <UnwiredPlatform_InstallDir>/
   Servers/UnwiredServer/Repository/Security/keystore.jks -
   deststoretype jks -deststorepass changeit -destkeypass
   changeit
   ```

   Even if the EIS administrator is using the native SAP public-key infrastructure (PKI) to generate certificates, you must still import them into the Unwired Server keystore. The certificate name, *SUPAUTH* and alias, *CERTALIAS* represent the type of package/client to be authenticated, for example:
   - TechnicalUser certificate with doectech alias – a DOE-C package client.
   - SAPUser certificate with SAPClient alias – a SAP or Web service MBO package client.
3. Create and import the SUPServer certificate into the Unwired Server keystore. For example:

   ```
   keytool -importkeystore -srckeystore SUPServer.p12 -
   srcstoretype pkcs12 -srcstorepass <supserverpass> -srcalias
   SUP -destkeystore <UnwiredPlatform_InstallDir>/Servers/
   UnwiredServer/Repository/Security/keystore.jks -
   deststoretype jks -deststorepass changeit -destkeypass
   changeit
   ```

   Note: (3a) You can create the SUPServer certificate using Java keytool commands, a third-party tool such as OPENSSL, or the signing authority used to create all SAP server certificates, in which case you need not import any other CA signing authority certificate into the Unwired Server truststore. However, if you create the SUPServer certificate with another CA signing authority, you must import that CA certificate into both the Unwired Server truststore, and into the SAP Server using the STRUST transaction.

4. Import the SUPServer certificate into SAP/DOE server using the STRUST transaction.

You can now configure your environment for mutual authentication and SSO, in which any client connecting to Unwired Server presents credentials, and a server certificate (SUPAUTH) is selected for Unwired Server to present to clients.

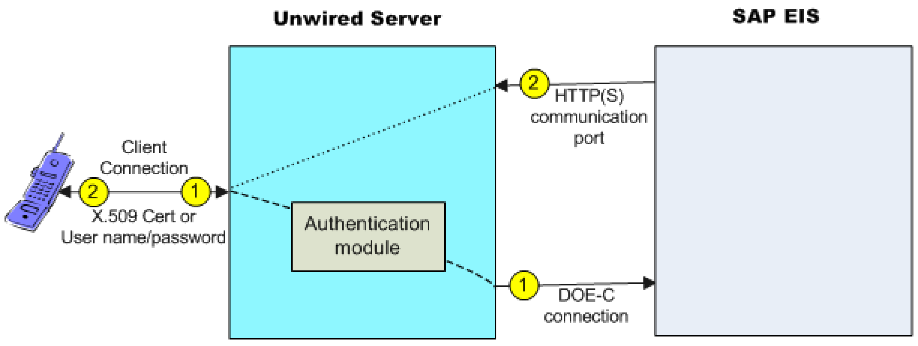
SAP Single Sign-on and DOE-C Package Overview

Understand how DOE-C packages fit in the Unwired Platform landscape, including how to secure communication paths and enable single sign-on (SSO) for these packages.

DOE-C is the connector from Unwired Server to SAP NetWeaver Mobile, which contains the DOE. Unwired WorkSpace is not used to create MBOs, generate code, create applications, or for deployment. Instead, in DOE-based mobile applications that run in the Unwired Platform environment:

- NetWeaver Mobile handles the data modeling for DOE-C connections.
- Field mappings, connection information, and other application- and package-specific information is defined in the ESDMA, for example the SAP CRM ESDMA, which is deployed to Unwired Server, and automatically converted into an Unwired Platform package by the ESDMA converter.
- DOE-C packages are message-based – NetWeaver Mobile is message-based, and performs queue handling, data caching, and is push-enabled to push data changes out to mobile devices through Unwired Server.

Unwired Server works as a pass-through gateway in the DOE/DOE-C configuration:



1. A DOE-C client application registers with Unwired Server and subscribes to message channels. Unwired Server remembers the push notification information/deviceID/ applicationID from the client, but forwards the subscription to DOE through the DOE-C

connection (HTTP(S)) to the DOE. When the client performs an operation, that operation flows through Unwired Server via this same connection to the DOE.

In an SSO configuration, the client provides credentials to Unwired Server (user name and password or X.509 user certificate) that are authenticated by the security configuration's authentication module (CertificateAuthenticationLoginModule for X.509 or HttpAuthenticationLoginModule for SSO2). Once authenticated by Unwired Server, and assuming that Unwired Server and the SAP Server have a secure communication path, SSO is enabled.

2. When application data changes in the SAP EIS and the DOE determines that a particular client has a subscription to that change, DOE connects to the Unwired Server HTTP(S) port and sends a message identifying the client, along with the message payload. Unwired Server looks up the client and queues a message. If the client is connected, the message is delivered immediately. If the client is offline, then Unwired Server attempts to send a push notification to the client (BES HTTP Push for Blackberry, APNS notification for iOS) to attempt to wake up the client and have it retrieve the messages.

   WindowsMobile does not have a separate push notification protocol, so Unwired Server waits for those clients to connect and retrieve their messages.

See also

- *Enabling Single Sign-on for DOE-C Packages* on page 23
- *Single Sign-on Authentication* on page 70

Enabling the DOE-C Connection

Configure the SAP Data Orchestration Engine Connector (DOE-C) connection pool between Unwired Server and the SAP EIS. This is the port on which Unwired Server communicates with the DOE, including forwarding subscriptions and allowing client operations to flow through to the DOE.

This type of connection is available in the list of connection templates only after a DOE-C package has been deployed to Unwired Server.

1. From Sybase Control Center, expand **Domains > *<DomainName>***, and select **Connections**.

   DomainName is the domain that contains the DOE-C package.
2. Select an existing connection pool, and set the property values required to enable an authenticated HTTPS connection to the DOE.

   If defining a security profile to implement mutual authentication with basic authentication, add the `certificateAlias` property, which overrides the technical user name and password fields. The technical user name and password fields can be empty, but only if `certificateAlias` is set. The specified certificate is extracted from the Unwired Server keystore and supplied to the DOE.
3. Select **Save**.

Deploying and Configuring DOE-C Packages

Unlike Mobile Workflow or MBO packages that use Sybase Control Center to deploy packages to Unwired Server, you must deploy the DOE-C package to specific domain using the DOE-C command line utility (CLU). Once deployed, the DOE-C package is visible and manageable from Sybase Control Center.

1. Start the command line utility console. See *Starting the Command Line Utility Console* in *System Administration*.
2. Deploy the DOE-C package. During deployment, you can set the domain and security configuration using the **setPackageSecurityConfiguration** command, or perform this task later from Sybase Control Center.

   See *Sybase SAP DOE Connector Command Line Utility* in the *System Administration* guide.

Next

Verify or set the security configuration for the domain or package.

See also

- *Security Configurations That Implement Single Sign-on Authentication* on page 92

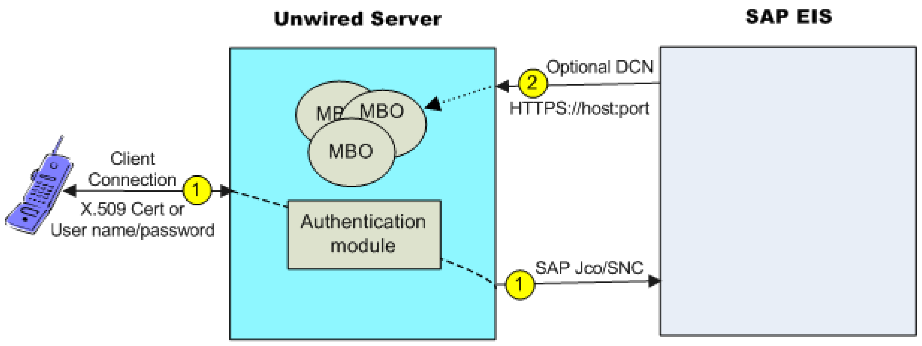
SAP Single Sign-on and Mobile Business Object Package Overview

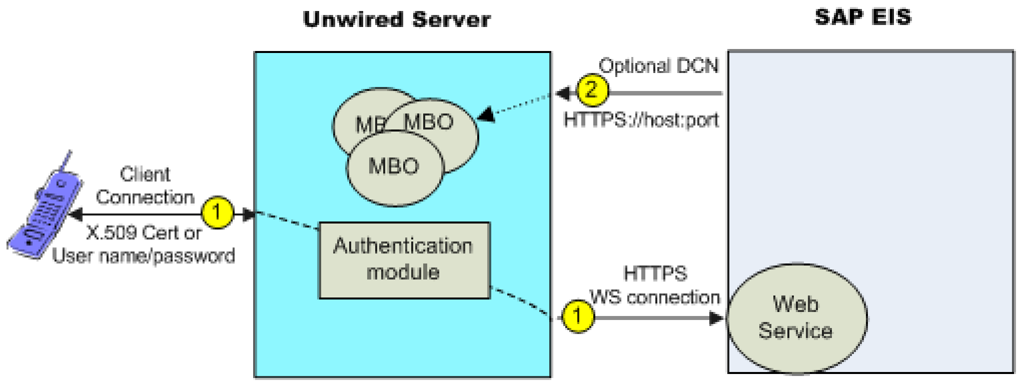
Understand how to secure communication ports and enable single sign-on (SSO) for packages that contain mobile business objects (MBOs) bound to an SAP enterprise information system (EIS).

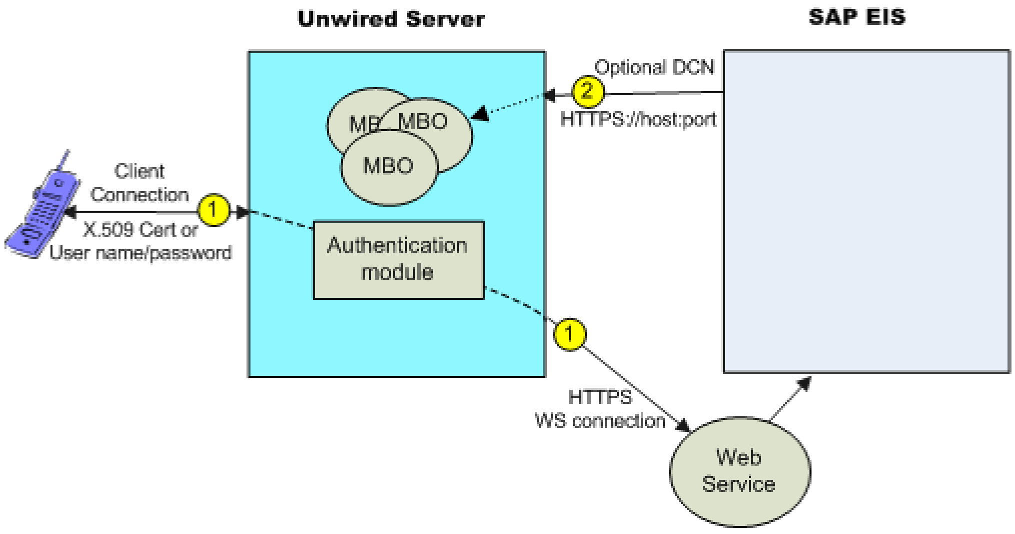
SAP MBOs bound directly to SAP BAPIs and RFCs, as well as SAP BAPIs exposed as Web services. Once deployed, Unwired Platform supports Java connector (JCo) connections and Secure Network Communications (SNC) for SAP MBOs, and HTTP(S) connections to Web services.

Once deployed, connection information, and other application- and package-specific information is maintained by Unwired Server. Unwired Server packages that contain SAP MBOs support message-based and replication-based applications and perform queue handling, data caching, and synchronization services.

Typical data flow for SAP MBO packages that use data change notification (DCN) as a refresh mechanism:


Unwired Server
SAP EIS
MB
MBO
MBO
Authentication module
Client Connection
1
X.509 Cert or User name/password
2
Optional DCN
HTTPS://host:port
1
SAP Jco/SNC


Unwired Server
SAP EIS
MB
MBO
MBO
Authentication module
Client Connection
1
X.509 Cert or User name/password
2
Optional DCN
HTTPS://host:port
1
HTTPS WS connection
Web Service



1. Data flows from Unwired Server to the EIS through a configured connection pool. For secure connections:
    - Jco – communicates with the SAP EIS using the SAP JCo proprietary communication protocol. Optionally use SNC if required for your installation.
    - Web service – communicates to the Web service host using HTTPS, whether the Web service is on the same server that hosts the SAP BAPIS/RFCs to which the Web service is bound, or a different server.

    In an SSO configuration, the client provides credentials to Unwired Server (username and password or X.509 user certificate) that are authenticated by the security configuration's authentication module (CertificateAuthenticationLoginModule for X.509 or HttpAuthenticationLoginModule for SSO2). Once authenticated by Unwired Server, and assuming that Unwired Server and the EIS have a secure communication path, SSO is enabled.
2. (Optional) Configure a data change notification (DCN) port if this is the data refresh policy for any of the MBOs within the package.

See also

- *Enabling Single Sign-on for Mobile Business Object Packages* on page 25
- *Single Sign-on Authentication* on page 70

Single Sign-on for SAP MBO Package Prerequisites

Before implementing SSO for SAP MBO packages, configure the MBOs so client credentials can be propagated to the EIS and, if enabling SSO for a Workflow application, add the appropriate starting point.

Configure the MBO. See these topics in *Sybase Unwired WorkSpace - Mobile Business Object Development*:

- *Propagating a Client's Credentials to the Back-end Data Source*
- *Configuring an MBO to Use an SAP Java Connector* – for SAP MBOs
- *Configuring an SAP Exposed Web Service MBO to Use Credentials*– for SAP function modules exposed as Web services

Configure the Workflow application to use SSO2 or X.509 credentials by adding and configuring a credential starting point for the workflow application. See *Configuring the Workflow Application to Use Credentials* in the *Developer Guide: Mobile Workflow Packages* for details.

Single Sign-on for SAP MBO Package Postrequisites

After configuring SSO for SAP MBO packages on Unwired Server, install certificates on the mobile device and test them.

- Workflow applications – see *Installing and Testing X.509 Certificates on Simulators and Mobile Devices* in the *Developer Guide for Mobile Workflow Packages*.
- Native applications – install and import X.509 certificates and use the Object API to select them for client connections. Refer to your platform's *Developer Guide* for details:
  - *Installing and Testing X.509 Certificates on Simulators and Mobile Devices*
  - *Single Sign-On With X.509 Certificate Related Object API*

Creating Connections and Connection Templates

Create a new connection or connection template that defines the properties needed to connect to a new data source.

1. In the left navigation pane, expand the **Domains** folder, and select the domain for which you want to create a new connection.
2. Select **Connections**.
3. In the right administration pane:
   - To create a new connection – select the **Connections** tab, and click **New**.
   - To create a new connection template – select the **Templates** tab, and click **New**.
4. Enter a unique **Connection pool name** or template name.
5. Select the **Connection pool type** or template type:
   - JDBC – choose this for most database connections.

- Proxy - choose this if you care connecting to the Online Data Proxy.
- WS – choose this if you are connecting to a Web Services (SOAP or REST) data source.
- SAP – choose this if you are connecting to an SAP (JCO) datasource.

6. Select the appropriate template for the data source target from the **Use template** menu. By default, several templates are installed with Unwired Platform; however, a production version of Unwired Server may have a different default template list.
7. Template default properties appear, along with any predefined values. You can customize the template, if required, by performing one of:
   - Editing existing property values – click the corresponding cell and change the value that appears.
   - Adding new properties – click the **<ADD NEW PROPERTY>** cell in the Property column and select the required property name. You can then set values for any new properties you add.

   Note: In a remote server environment, if you edit the sampledb Server Name property, you must specify the remote IP number or server name. Using the value "localhost" causes cluster synchronization to fail.
8. Test the values you have configured by clicking **Test Connection**. If the test fails, either values you have configured are incorrect, or the data source target is unavailable. Evaluate both possibilities and try again.
9. 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.

Configuring an SAP Java Connector With SNC

Create a Java Connection (JCo) to an SAP Server in Unwired Server from Sybase Control Center where SNC is required.

Prerequisites

Start Unwired Server services and log in to Sybase Control Center as the administrator, and download and install the SAP cryptographic libraries.

Task

The SAP JCo connection provides access for various client types, including those that use SSO2 tokens and X.509 certificates.

1. Expand the cluster, expand the **Domains** folder, expand the domain to which the package is to be deployed, and select **Connections**.
2. Select the **Connections** tab and click **New**. Name the connection pool `SAP Server`, select **SAP** as the Connection pool type, select the **SAP template**, and enter appropriate

properties for the SAP enterprise information system (EIS) to which you are connecting. For example:

Alternatively, if you require a template with these SNC properties prepopulated for future convenience, create a new template for SNC-enabled SAP connections, and then use that template for these properties.

- Language (jco.client.lang) = `EN`
- Logon User (jco.client.user)=`snctest`
- Password (jco.client.password) =`********` (snctest user password)
- Host name (jco.client.ashost) = `sap-doe-vm1.sybase.com`
- System number (jco.client.sysnr) = `00`
- SNC mode (jco.client.snc_mode) = `1`
- SNC name (jco.snc_myname) = `p:CN=SNCTEST, O=Sybase, L=Dublin, SP=California, C=US`
- SNC service library path (jco.client.snc_lib) = `C:/sapcryptolib/sapcrypto.dll` (the location of the cryptographic library)
- Client number (jco.client.client) = `100`
- SNC partner (jco.client.snc_partnername) = `p:CN=sap-doe-vm1, OU=SUP, O=Sybase, C=US`
- SNC level (jco.client.snc_qop) = `1`

3. Click **Test Connection** to verify access to the SAP server, and click **OK**.

Generating and Installing a PSE Certificate on Unwired Server

Generate a PSE certificate on Unwired Server to use in testing connections with SAP Systems when using the SAP Cryptographic Library to secure the connection using Secure Network Communications (SNC).

Prerequisites

Download and install the SAP Cryptographic Library.

Task

These instructions describe how to generate an X.509 certificate for testing SAP JCo and single sign-on with SNC only. In a production environment, a different entity controls certificate management. For example, an SAP system administrator controls certificate generation and management for his or her particular environment, including maintaining the certificate list in a Personal Security Environment (PSE) with trust manager.

Note: When the CertificateAuthenticationLoginModule gets a certificate from a client, it can optionally validate that it is a trusted certificate. The easiest way to support validation is to import the CA certificate into the `<UnwiredPlatform_InstallDir>/Servers/UnwiredServer/Repository/Security/truststore.jks` file, which is the default Unwired Server truststore.

Use the SAPGENPSE utility to create a PSE certificate to use for testing. See *http://help.sap.com/saphelp_nw04s/helpdata/en/a6/f19a3dc0d82453e10000000a114084/content.htm*. The basic steps are:

1. Generate the certificate from the SAP Cryptograhpic Library directory. For example,`C:\sapcryptolib`:

   ```
   sapgenpse get_pse <additional_options> -p <PSE_Name> –r
   <cert_req_file_name> -x <PIN> <Distinguished_Name>
   ```

2. Copy the PSE certificate (for example, `SNCTEST.pse`) to the location of your installed SAP Cryptographic Library. For example, `C:\sapcryptolib`.
3. Generate a credential file (cred_v2) from the `C:\sapcryptolib` directory:

   ```
   sapgenpse seclogin -p SNCTEST.pse -O DOMAIN\user -x
   password
   ```

   Note: The user that generates the credential file must have the same user name as the process (that is, either `mlserv32.dll` or `eclipse.exe`) under which the Unwired Platform service runs. The user must also be user of the domain as determined with the `-O` *`DOMAIN`*`\`*`user`* flag.

SAP Java Connector Properties

Configure SAP Java Connector (JCo) connection properties.

For a comprehensive list of SAP JCo properties you can use to create an instance of a client connection to a remote SAP system, see *http://help.sap.com/javadocs/NW04/current/jc/com/sap/mw/jco/JCO.html#createClient(java.util.Properties)*.

Table 4. General connection parameters

Name	Description	Supported values
Client Number	Specifies the SAP client.	Three-digit client number; preserve leading zeros if they appear in the number
Logon User	Specifies the login user ID.	User name for logging in to the SAP system If using X.509 certificate authentication, remove the JCo properties `jco.client.passwd` and `jco.client.user` defined for the SAP connection profile in Sybase Control Center (SCC).
Password	Specifies the login password.	Password for logging in to the SAP system

Name	Description	Supported values
Language	Specifies a login language.	ISO two-character language code (for example, EN, DE, FR), or SAP-specific single-character language code. As a result, only the first two characters are ever used, even if a longer string is entered. The default is EN.
System Number	Indicates the SAP system number.	SAP system number
Host Name	Identifies the SAP application server.	Host name of a specific SAP application server
Message Server	Identifies the SAP message server.	Host name of the message server
Gateway Host	Identifies the SAP gateway host.	Host name of the SAP gateway Example: `GWHOST=hs0311`
Gateway Service	Identifies the SAP gateway service.	Service name of the SAP gateway Example: `GWSERV=sapgw53`
R/3 Name	Specifies R/3 name.	Name of the SAP system
Server Group	Identifies the group of SAP application servers.	Group name of the application servers
External Server Program	Identifies the program ID of the external server program.	Path and name of the external RFC server program, or program ID of a registered RFC server program Example: `TPNAME=/sap/srfcserv`
External Server Program Host	Identifies the host of the external server program. This information determines whether the RFC client connects to an RFC server started by the SAP gateway or to an already registered RFC server. **Note:** If the gateway host and external server program host are different, make sure that the SAP gateway has access to start the server program through a remote shell.	Host name of the external RFC server program Example: `TPHOST=hs0311`

Name	Description	Supported values
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
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.

Name	Description	Supported values
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 RfcGet-PartnerSSOTicket() 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	Unwired Platform sets this property when a client uses an SAP Cookie Version 2 (SSO2) as the login credential .	Unwired Platform uses the user and password properties to pass these values: User: $MYSAPSSO2$ Password: Base64-encoded ticket
Use X509	Unwired Platform sets this property when a client uses an X509 certificate as the login credential.	Unwired Platform uses the user and password properties as follows to pass certificate values: User: $X509CERT$ Password: Base64-encoded ticket
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

Name	Description	Supported values
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)
SNC Level	Specifies the secure network connection security level.	1: digital signature 2: digital signature and encryption 3: digital signature, encryption, and user authentication 8: default value defined by backend system 9: maximum value that the current security product supports
SNC Name	Indicates the secure network connection name. This property overrides the default secure network connection partner.	Token or identifier representing the external RFC program
SNC Service Lib Path	Identifies the path to the SAP crytpographic library that provides secure network connection service.	Full path and name of third-party security library. You must download and install the library from the SAP Service Marketplace.
R/2 Destination	Identifies a configured R/2 system defined in the sideinfo configuration.	
Logon ID	Defines the string for SAPLOGON on 32-bit Windows.	String key to read parameters from the saplogon.ini file created by the SAPLogon GUI program on Windows

Name	Description	Supported values
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.	

Web Services Properties

Configure connection properties for the Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) architectures.

Name	Description	Supported Values
Password	Specifies the password for HTTP basic authentication, if applicable.	Password
Address	Specifies a different URL than the port address indicated in the WSDL document at design time.	HTTP URL address of the Web service
User	Specifies the user name for HTTP basic authentication, if applicable.	User name
Certificate Alias	Sets the alias for the Unwired Platform keystore entry that contains the X.509 certificate for Unwired Server's SSL peer identity. If you do not set a value, mutual authentication for SSL is not used when connecting to the Web service.	Use the alias of a certificate stored in the Unwired Server certificate keystore.

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

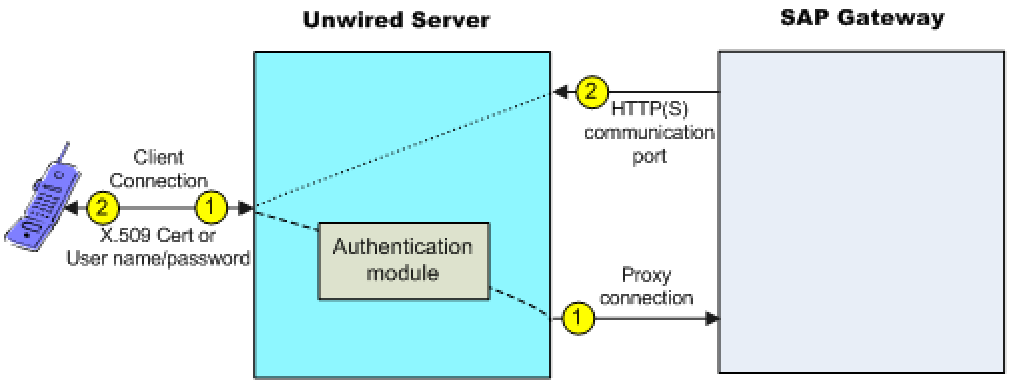
SAP Single Sign-on and Online Data Proxy Overview

Understand how OData applications fit in the Unwired Platform landscape and learn how to secure communication paths and enable single sign-on (SSO) for these applications.

The proxy connector is the online data proxy (ODP) connector between OData applications and the SAP Gateway, and uses an HTTP(S) connection from Unwired Server to the SAP Gateway. A separate HTTP(S) port is used by the SAP Gateway to push changes through Unwired Server to the OData application. Unwired WorkSpace is not used to create MBOs, generate code, create applications, or for deployment. Instead, in OData-based mobile applications that run in Unwired Server:

- Applications are developed using the OData SDK.
- The SAP Gateway/enterprise information system (EIS) is responsible for data federation and content management.
- OData applications are message based – the SAP Gateway performs queue handling, data caching, and is push-enabled to push data changes out to Unwired Server, which in turn pushes these changes to the physical devices.

Unwired Server acts as a pass-through server for OData-based applications:



1. An OData client application registers with Unwired Server and subscribes to push notifications from the SAP Gateway. Unwired Server forwards the subscription request to the SAP Gateway. The SAP Gateway stores the subscription request for the collection with the push delivery address (HTTP(S) SSL Port).

   In an SSO configuration, the client provides credentials to Unwired Server (user name and password or X.509 user certificate) that are authenticated by the security configuration's authentication module (CertificateAuthenticationLoginModule for X.509 or HttpAuthenticationLoginModule for SSO2). Once authenticated by Unwired Server, and assuming that Unwired Server and the SAP Gateway have a secure communication path, SSO is enabled.
2. When application data changes in SAP and determines that a particular client has a subscription to that change, the Gateway connects to the Unwired Server HTTP(S) port and sends a message identifying the client, along with the message payload. Unwired Server looks up the client and queues the message. If the client is connected, the message is delivered immediately. If the client is offline, then Unwired Server attempts to send a push notification to the client (BES HTTP Push for Blackberry, APNS notification for iOS) to attempt to wake up the client and have it retrieve the messages.

See also

- *Single Sign-on Authentication* on page 70
- *Enabling Single Sign-on for OData Applications* on page 24

Preparing the SAP Gateway

Configure the SAP Gateway to push OData application data to Unwired Server, including configuring the RFC destination for HTTPS on the Gateway.

1. Log on to a Gateway system and go to transaction **sm59**.
2. Create a RFC connection of type **G**.

3. Enter the domain name of Unwired Server (CN of the Unwired Server certificate) in the **Target Host** field .
4. Enter `/GWC/SUPNotification` in the **Path Prefix** field.
5. Enter 8004 in the **Service No.** field.
6. Select the **Logon & Security** tab.
7. Under **Security Options**, click the SSL **Active** option.
8. Select **Default SSL Client (Standard)** from the **SSL Certificate** drop-down list.
9. Click **Save**, then **Connection Test**.

   The test should be successful, a HTTP OK success message displays.

   Note: Change the push endpoint in the proxy property of the application templates to https://<domain name of the SUP server>:<SSL Port>/GWC/SUPNotification/. In this example, 8004 is the SSL port to which the Gateway pushes data changes:

   ```
   https://inln50089324a.dhcp.blrl.sap.corp:8004/GWC/
   SUPNotification/
   ```

Preparing Your SAP Environment for Single Sign-on

Verify that the SAP® enterprise information system (EIS) is configured correctly to accept SSO connections from Unwired Server.

The general steps for enabling SAP systems to communicate with Unwired Server over secure communication paths are:

1. Set all parameters for the type of credentials accepted by the server:
   - SSO2 token – verify everything is set properly with the SSO2 transaction.
   - X.509 certificate – set up, import, and verify certificates using the Trust Manager (transaction STRUST).
2. Use the ICM configuration utility to enable the ICM HTTPS port.
3. Set the type of authentication to enable over HTTPS:
   - Server authentication only – the server expects the client to authenticate itself using basic authentication, not SSL
   - Client authentication only – the server requires the client to send authentication information using SSL certificates. The ABAP stack supports both options. Configure the server to use SSL with client authentication by setting the ICM/HTTPS/verify_client parameter:
     - 0 – do not use certificates.
     - 1 – allow certificates (default).
     - 2 – require certificates.
4. Use the Trust Manager (transaction STRUST) for each PSE (SSL server PSE and SSL client PSE) to make the server's digitally signed public key certificates available. Use a public key infrastructure (PKI) to get the certificates signed and in the SAP system.

   There are no SSO access restrictions for MBO data that span multiple SAP servers.

See SAP product documentation at *http://help.sap.com/saphelp_aii710/helpdata/en/49/23501ebf5a1902e10000000a42189c/frameset.htm* for information about the SAP Trust Manager.

5. To enable secure communication, Unwired Server and the SAP server it communicates with must exchange valid CA X.509 certificates. Deploy these certificates, which are used during the SSL handshake with the SAP server into the Unwired Server truststore.
6. The user identification (distinguished name), specified in the certificate must map to a valid user ID in the AS ABAP, which is maintained by the SM30 view (VUSREXTID).

See *Configuring the AS ABAP for Supporting SSL* at *http://help.sap.com/saphelp_aii710/helpdata/en/49/23501ebf5a1902e10000000a42189c/frameset.htm*

Security Configurations That Implement Single Sign-on Authentication

Use the CertificateAuthenticationLoginModule authentication module to implement X.509 authentication or HttpAuthenticationLoginModule to implement SSO2.

Creating and Assigning a Security Configuration That Uses SSO2 Tokens

Create a new security configuration, assign the HttpAuthenticationLoginModule authentication provider to it, and assign the security configuration to an Unwired Server domain or package.

The HttpAuthenticationLoginModule authentication provider supports SSO2 token logins to SAP systems through JCo and Web service connections, DOE-C packages, and other packages that require token authentication.

1. Create the new security configuration:
   a) From Sybase Control Center, select **Security**.
   b) Select the **General** tab, click **New**, and enter a name for the new security configuration, for example, `SAPSSOSECADMIN`. Click **OK**.
2. Configure the SAP EIS portal:
   a) Apply SAP Note 1250795 to the portal server. This is required to get the HTTP challenge pop-up window.
   b) Verify the SAP EIS URL configured as the Unwired Server SAP Server URL property is an URL with a challenge popup window, not just a generic portal URL.
   c) Maintain the URL and control flag security configuration parameters, which are the only required parameters.
3. Configure the new security configuration:
   a) Select the **SAPSSOSECADMIN** security configuration.
   b) Select the **Authentication** tab.
   c) Click **New** and select **HttpAuthenticationLoginModule** as the authentication provider. Set the SAP server URL, the SSO cookie name (typically set to `MYSAPSSO2`), and other properties as appropriate for the connection.

4. Select the **General** tab, and click **Validate** to confirm that Unwired Server accepts the new security configuration.
   A message indicating the success of the validation appears above the menu bar.
5. Click **Apply** to save changes to the security configuration, and apply them across Unwired Server.
6. Assign the SAPSSOSECADMIN security configuration to the domain to which SSO packages are being deployed.
   a) Click **Domains > *DomainName* > Security** .
   b) Click **Assign**.
   c) Select **SAPSSOSECADMIN** and click **OK**.
7. If any other security configurations have been assigned to this SSO domain, Sybase suggests that you unassign them.

   However, many deployments of Unwired Platform do mix SSO and non-SSO MBOs or operations in the same package. There are certain operations that are not sensitive and do not require the overhead of setting up the SSO connection to the backend. Some packages may even perform DCNs, and the DCN user would not be part of the SSO-enabled login module. If you do authenticate a user against a non-SSO login module and then attempt to perform an SSO-enabled operation, then the credentials are sent to the backend, which may not be desired.

Creating and Assigning a Security Configuration That Uses X.509 Credentials

Create a new security configuration, assign the CertificateAuthenticationLoginModule authentication provider to it, and assign the security configuration to an Unwired Server domain or package.

The CertificateAuthenticationLoginModule authentication provider supports X.509 certificate logins to SAP systems through JCo, DOE-C, Online Data Proxy, and Web service connections. You can assign security configurations to domains, packages, or applications.

1. Create the new security configuration:
   a) From Sybase Control Center, select **Security**.
   b) Select the **General** tab, click **New**, and enter a name for the new security configuration, for example, `X509SECADMINCERT`. Click **OK**.
2. Configure the new security configuration:
   a) Expand the Security folder.
   b) Select the **X509SECADMINCERT** security configuration.
   c) Select **Authentication**.
   d) Select **New**.
   e) Select **com.sybase.security.core.CertificateAuthenticationLoginModule** as the Authentication provider.
   f) Click **OK** to accept the default settings, or modify any of these settings as required:

    - Click **<Add New Property>**, select **Validate Certificate Path** and set the value to **true**.
    - If more than one truststore is defined in Unwired Server, click **<Add New Property>**, select **Trusted Certificate Store** and set the value to the location of the Java truststore that contains the Unwired Server trusted CA certificates. Otherwise, the default Unwired Server truststore is used.
    - If you change the default password for the truststore, click **<Add New Property>**, select **Trusted Certificate Store Password** and set the value of the truststore password.

   g) Click **OK**.

3. Select the **General** tab, select **Validate**, then **Apply**.
4. Assign the X509SECADMINCERT security configuration to an Unwired Server domain. This example uses the default domain, but you can specify any domain to which the package is deployed:
   a) Click **Domains > *DomainName* > Security** .
   b) Click **Assign**.
   c) Select **X509SECADMINCERT** and click **OK**.
5. If any other security configurations have been assigned to this SSO domain, Sybase suggests that you unassign them.

   However, many deployments of Unwired Platform do mix SSO and non-SSO MBOs or operations in the same package. There are certain operations that are not sensitive and do not require the overhead of setting up the SSO connection to the backend. Some packages may even perform DCNs, and the DCN user would not be part of the SSO-enabled login module. If you do authenticate a user against a non-SSO login module and then attempt to perform an SSO-enabled operation, then the credentials are sent to the backend, which may not be desired.

Creating Security Profiles to Enable Mutual Authentication for SAP

Create security profiles and associate them with X.509 server certificates that can be used to establish secure connections between a client, Unwired Server, and the SAP EIS.

Prerequisites

- Your SAP system must be configured for HTTPS mutual authentication
- Import the third party's private-key certificate used by Unwired Server to mutually authenticate the client into the Unwired Server keystore:
  - *SUPServer* certificate – represents the certificate used to secure an HTTPS connection between Unwired Server and SAP Server or other enterprise information system (EIS), where data and information flow from Unwired Server to the EIS, which could be a DOE-C, Web Service, or Proxy connection.
  - *SAPServer* certificate – represents the certificate used to secure the communication path between the SAP Server or EIS and Unwired Server, where data and information flow from the EIS to Unwired Server on an HTTPS port (8001, 8002, and so on), which

are made available to the EIS for pushing data to Unwired Server. For SAP Servers, this could be NetWeaver/DOE (TechnicalUser), or the SAP Gateway.

Task

To secure connections, create two new security profiles: one for the SAP gateway and one for Unwired Server. If you imported the user and CA certificates into keystore or truststore locations other than the default, make sure the paths and passwords reflect them.

1. In the Sybase Control Center navigation pane, click **Servers > *ServerName* > Server Configuration**.
2. From the **General** tab, click **SSL Configuration**.
3. Select **<ADD NEW SECURITY PROFILE>** and create a security profile for SAP servers:
   - Security profile name – for example, `TechnicalUser`for NetWeaver/DOE connections or `Proxy` for SAP Gateway connections.
   - Certificate alias – the case sensitive certificate alias you defined when you imported the certificate into the keystore. For example, `doetech`, `proxy` (or whatever value you set the alias to using the **keytool -alias** option).
   - Authentication – `strong_mutual`
4. Select **<ADD NEW SECURITY PROFILE>** and create an Unwired Server security profile:
   - Security profile name – `SUPServer`.
   - Certificate aias – `SUP` (or whatever value you set the alias to using the **keytool -alias** option).
   - Authentication– `strong_mutual`.
5. Restart Unwired Server.

Enabling the HTTPS Port and Assigning the Unwired Server Security profile

Enable an HTTPS port for secure communication between Unwired Server and the SAP EIS.

For DOE-C and SAP Gateway, this is the port to which the DOE or Gateway connects to Unwired Server and sends a message identifying the client, along with the message payload.

1. In the Sybase Control Center navigation pane, click **Servers > *ServerName* > Server Configuration**.
2. From the **General** tab, click **Communication Ports**.
3. Expand **Show secure data change notification ports**.
4. Select a port number and enter:
   - Status – `enabled`.
   - SSL Security Profile – `SUPServer` or whatever you named the security profile associated with the Unwired Server user certificate.

Note: You can add a new HTTPS port if you do not want to use 8001 or 8002. For Gateway connections, the port must match that of the port you defined in the Gateway, for example 8004. See *Preparing the SAP Gateway*.

5. Select **Save**.
6. Restart Unwired Server.

See also

- *Preparing Your SAP Environment for Single Sign-on* on page 91

Distributing Single Sign-on Related Files in an Unwired Server Cluster

Place required files in the appropriate primary Unwired Server subdirectory so they are distributed to all Unwired Servers within the cluster during cluster synchronization.

Any changes to a named security configuration affect the cluster and trigger a cluster synchronization, which automatically zips the files in the primary Unwired Server `CSI` subdirectory and distributes them to the other servers in the cluster. Copy all certificate and other security-related files to the `CSI` subdirectory.

The provider configuration information, which includes the server certificate file name and location, must be the same on all cluster nodes. The same is true for the cryptographic DLLs and certificate files for SSO using X509.

1. On the primary server in the cluster, put any SAP certificate files or truststores into the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\CSI\conf` directory.

   Use system properties to specify the full path and location of the file in the configuration so they can be accessed from different servers within the cluster if installation directories are different from that of the primary server. For example:

   ```
   ${djc.home}/Repository/CSI/conf/
   SNCTEST.pse
   ```

   For X.509 CertificateAuthenticationLoginModule, if the *ValidateCertificatePath* is set to true, the default, the CA certificate (or one of its parents) must be installed in the truststore for each server.

   Note: Unwired Server truststore and keystore files:
   - `<UnwiredPlatform_InstallDir\Servers\UnwiredServer\Repository\Security\truststore.jks` – is the Unwired Server trust store that contains CA (or parent) certificates. Unwired Server trusts all CA or parent certificates in `truststore.jks`.
   - `<UnwiredPlatform_InstallDir\Servers\UnwiredServer\Repository\Security\keystore.jks` – contains client certificates only.

The CertificateAuthenticationLoginModule also has `Trusted Certificate Store*` and `Store Password` properties which you can to keep the module out of the default Unwired Server trust store. You must first:

a) Use **keytool** to put the CA certificate into a new keystore.
b) Put the keystore into the `Repository\CSI\conf` subdirectory.
c) Include the path in the `Trusted Certificate Store` property.

2. From Sybase Control Center, add the login module.
3. Restart all Unwired Servers within the cluster.

Enabling CRLs

Identify the certificate revocation lists (CRLs) that define a list of digital certificates which have been revoked. Revoked certificates should not give the Unwired Platform device user access to the Unwired Server runtime.

Administrators can configure certificate revocation lists (CRLs) to check if any of the certificates in the path are revoked. A series of URI's define the CRL location.

1. Using Sybase Control Center, open the CertificateAuthenticationLoginModule use by your security configuration.
2. For the CRL property, define one or more URIs. If using multiple URIs, each must be indexed.

   The index number used determines the order in which CLRs are checked. This example uses two URI, each indexed accordingly so that the Verisign CRL comes first.

   ```
   crl.1.uri=http://crl.verisign.com/
   ThawtePersonalFreemailIssuingCA.crl
   crl.2.uri=http://crl-server/
   ```

Authentication Cache Timeouts

Set a cache timeout value to cache user or administrator authentication credentials, which improves runtime performance.

Set timeout properties to avoid repeatedly reauthenticating users— a benefit of particular interest for device clients that receive separately authenticated messages. If a user logs in successfully, he or she can reauthenticate with the same credentials without validating them against a security repository. However, if the user provides a user name or password that is different from the ones cached, Unwired Server delegates the authentication request to the security repository.

This property affects only authentication results:

- The successful authentication result is cached.
- If authentication passes, user roles are assigned.

Authorization results and failed authentication results are not cached.

For example, if an MBO is protected by "LogicalRoleA", and the security configuration that MBO is deployed in has a role mapping to "PhysicalRoleA", each time a user tries to access this MBO, the provider checks to see if they are in PhysicalRoleA based on cached role membership from the original authentication. It does not check the security repository each time thereafter.

By default, the cache timeout value is 3600 (seconds). This value is enabled whether the property exists or not. You can change this value by configuring a new value for the property in Sybase Control Center for the appropriate security configuration. Or, you can set the value to 0, to restrict access and force reauthentication.

See also

- *Enabling Authentication Caching and Reducing Log Levels* on page 41
- *Disabling Authentication Caching and Increasing Log Levels* on page 40

Stacking Providers and Combining Authentication Results

(Not applicable to Online Data Proxy) Optionally, implement multiple login modules to provide a security solution that meets complex security requirements. Sybase recommends provider stacking as a means of eliciting more precise results, especially for production environment that require different authentications schemes for administrators, DCN, SSO, and so on.

Stacking is implemented with a controlFlag attribute that controls overall behavior when you enable multiple providers. Set the controlFlag on a specific provider to refine how results are processed.

For example, say your administrative users (supAdmin in a default installation) are not also users in an EIS system like SAP. However, if they are authenticated with just the default security configuration, they cannot also authenticate to the SAPSSOTokenLoginModule used for SSO2Token retrieval. In this case, you would stack a second login modules with a controlFlag=sufficient login module for your administrative users.

Or, in a custom security configuration (recommended), you may also find that you are using a technical user for DCN who is also not an SAP user. This technical user does not need SSO because they will not need to access data. However, the technical user still needs to be authenticated by Unwired Server. In this case, you can also stack another login module so this DCN user can login.

1. Use Sybase Control Center to create a security configuration and add multiple providers as required for authentication.
2. Order multiple providers by selecting a login module and using the up or down arrows at to place the provider correctly in the list.

   The order of the list determines the order in which authentication results are evaluated.
3. 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.

4. Click **Save**.
5. Select the **General** tab, and click **Apply**.

For example, say you have sorted these login modules in this order and used these controlFlag values:

- LDAP (required)
- NT Login (sufficient)
- SSO Token (requisite)
- Certificate (optional)

The results are processed as indicated in this table:

Provider	Authentication Status							
LDAP	pass	pass	pass	pass	fail	fail	fail	fail
NT Login	pass	fail	fail	fail	pass	fail	fail	fail
SSO Token	*	pass	pass	fail	*	pass	pass	fail
Certificate	*	pass	fail	*	*	pass	fail	*
Overall result	pass	pass	pass	fail	fail	fail	fail	fail

See also

- *LDAP Security Provider* on page 37
- *LDAP Configuration Properties* on page 151
- *Assigning Providers to a Security Configuration* on page 61
- *Assigning Security Configurations to Domains, Packages, or Applications* on page 62

controlFlag Attribute Values

(Not applicable to Online Data Proxy) The Sybase implementation uses the same controlFlag values and definitions as those defined in the JAAS specification.

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

Control Flag Value	Description
(Default) required	The LoginModule is required. Authentication proceeds down the LoginModule list.
requisite	The LoginModule is required. Subsequent behavior depends on the authentication result: • If authentication succeeds, authentication continues down the LoginModule list. • If authentication fails, control returns immediately to the application (authentication does not proceed down the LoginModule list).
sufficient	The LoginModule is not required. Subsequent behavior depends on the authentication result: • If authentication succeeds, control returns immediately to the application (authentication does not proceed down the LoginModule list). • If authentication fails, authentication continues down the LoginModule list.
optional	The LoginModule is not required. Irrespective of success or failure, authentication proceeds down the LoginModule list.

Example

Providers are listed in this order and with these controlFlag:

1. CertificateAuthenticationLoginModule (sufficient)
2. LDAP (optional)
3. NativeOS (sufficient)

A client doing certificate authentication (for example, X.509 SSO to SAP) can authenticate immediately. Subsequent modules are not called, because they are not required. If there are regular username/password credentials, they go to LDAP, which may authenticate them, and set them up with roles from the LDAP groups they belong to. Then NativeOS is invoked, and if that also succeeds, Unwired Platform picks up roles based on the Windows groups they are in.

Stacking LoginModules in SSO Configurations

(Not applicable to Online Data Proxy) Use LoginModule stacking to enable role-based authorization for MBOs and data change notifications (DCNs).

1. *Retrieving Roles for Subjects Authenticating to Single Sign-on Enabled Login Modules*

The CertificateAuthenticationLoginModule does not extract role information. If MBOs and MBO operations have roles assigned, stack login modules to get roles for the user.

2. *Stacking Login Modules to Allow for DCN in Packages Using SSO Login Modules*

   All DCN operations require the "SUP DCN User" logical role in the named security configuration (role mapping applies).

Retrieving Roles for Subjects Authenticating to Single Sign-on Enabled Login Modules

The CertificateAuthenticationLoginModule does not extract role information. If MBOs and MBO operations have roles assigned, stack login modules to get roles for the user.

1. HttpAuthenticationLoginModule – username and password credentials are supplied by the user. If these credentials go to an LDAP/AD EIS, add an LDAPAuthorizer with appropriate properties to look up the LDAP subject and retrieve LDAP groups as roles. You can also use the csi-userrole authorizer; but role-mapping maintenance is onerous with a large user base.
2. CertificateAuthenticationLoginModule – use the csi-userrole provider to map logical roles to physical roles named user:*subject* where *subject* matches the common name (CN=*xxx*) from the X.509 certificate.

   See *Configuring an LDAP Authentication Module* in *Sybase Control Center online help*.

Stacking Login Modules to Allow for DCN in Packages Using SSO Login Modules

All DCN operations require the "SUP DCN User" logical role in the named security configuration (role mapping applies).

An additional login module with authorization, that can assign a physical role, is required. Module stacking authenticates DCN users, and grants them the DCN role. Ordering of modules and control flag settings in the security configuration can vary. For example:

1. HttpAuthenticationLoginModule – the CertificateAuthenticationLoginModule is first in the list with the controlFlag set to "sufficient". If authentication succeeds, no other Login Modules are called unless their controlFlags are set to "required".
2. If CertificateAuthenticationLoginModule is used to authenticate mobile users, stack another login module for the DCN user because the DCN user does not have a "certblob" credential with the X.509 certificate.
   a) Set the CertificateAuthenticationLoginModule's controlFlag to "sufficient", and order it first in the stack. This sequence allows normal device users to authenticate quickly.
   b) Choose any other username/password-based login module to stack with its controlFlag set to "optional" (or sufficient). If this login module does not include an authorizer that can retrieve roles, use the csi-userrole provider.

      For example, if a DCN includes one DCN technical user, it requires only one role mapping, from "SUP DCN User" to user:dcnTechnicalUser.

Security Provider Issues

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

If 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. Search for *Logs* in the *Sybase Control Center online help*. If you are still experiencing issues, look for problems and solutions in the *Troubleshooting* guide.

Encrypting Synchronization for Replication Payloads

(Not applicable to Online Data Proxy) By default, the Unwired Server replication listener is configured to use TLS for end-to-end encryption (E2EE) on HTTP and HTTPS ports, and SSL for encryption on HTTPS ports.

If you do not require both TLS and SSL, you can disable either of them by modifying the replication synchronization listener in Sybase Control Center.

Once the listener is configured, applications must then connect to the Relay Server port and use an appropriate protocol:

- The administrator can define an application template in Sybase Control Center.
- The developer can call the Object API to set the E2EE and HTTPS items in the synchronization profile.

When devices onboard with Unwired Platform, clients receive their initial configuration settings from the application template and public keys, and HTTPS public certificate files are provisioned as part of these configuration settings.

1. *Understanding Encryption Requirements and Limitations*

   TLS encryption is the recommended method for closing the WAP gap that may exist with synchronization.
2. *Changing Installed Certificates Used for Encryption*

   Unwired Server includes default certificates for all listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.
3. *Modifying Default Synchronization Listener Properties with Production Values*

   Once you have determined the degree of secure communication you require, you may need to modify default synchronization listener property values to disable one or more ports or synchronization protocols.
4. *Encryption Postrequisites*

With the replication synchronization listener configured, ensure that the client application is provisioned with required artifacts and has connections configured accordingly.

See also

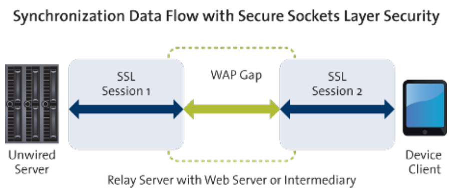
- *Provisioning Security Artifacts* on page 132
- *Unwired Server and Device Application Communications* on page 5
- *Certificate Creation (createcert) Utility* on page 183
- *Key Creation (createkey) Utility* on page 186
- *End-to-End-Encryption with TLS* on page 103

End-to-End-Encryption with TLS

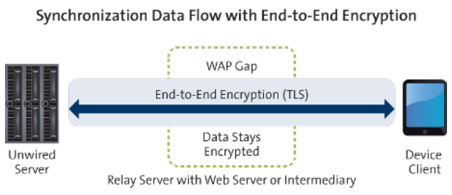
Wireless Application Protocol (WAP) has an issue commonly referred to as the WAP gap. You can Secure client/server synchronization with transport-layer security (TLS) to prevent WAP gap compromises.

TLS takes advantage of digital certificates and public-key cryptography to enable encryption, tamper detection, and certificate-based authentication for entity state replication environments.

A WAP gap breaks end-to-end communication data privacy. TLS encryption closes the WAP gap to ensure the synchronization stream is protected and secure. This diagram shows a traditional SSL synchronization stream passing through the WAP gap, where one SSL session encrypts the device-to-Relay Server stream and the other encrypts the Relay Server-to-Unwired Server stream:


Synchronization Data Flow with Secure Sockets Layer Security
SSL Session 1
WAP Gap
SSL Session 2
Unwired Server
Device Client
Relay Server with Web Server or Intermediary

Once you enable TLS to encrypt the entire synchronization data stream, you avoid passing unencrypted data through this WAP gap:


Synchronization Data Flow with End-to-End Encryption
WAP Gap
End-to-End Encryption (TLS)
Data Stays Encrypted
Unwired Server
Device Client
Relay Server with Web Server or Intermediary

Note: End-to-end encryption for Unwired Platform supports RSA encryption only.

See also

- *Encrypting Synchronization for Replication Payloads* on page 102

Understanding Encryption Requirements and Limitations

TLS encryption is the recommended method for closing the WAP gap that may exist with synchronization.

Encryption requires:

1. Keys and certificates: By default, synchronization is automatically configured for end-to-end encryption (E2EE) and uses the HTTPS certificates for mutual authentication. The default configuration uses the default key and certificate pairs. You must exchange these defaults with production-ready ones. For example:
    - For E2EE with TLS, generate the E2EE private key and public key at the same time; the public key is provisioned to the device client.
    - For HTTPS with SSL, generate the HTTPS server identity certificate and server public certificate at the same time; the HTTPS public certificate is provisioned to the device client.
2. Encryption protocols: You can choose the degree of security you require. Unwired Server can support multiple protocols.
    - Recommended – use HTTP (either with or without E2EE/TLS) on port 2480.
    - Use HTTPS (either just SSL, or just E2EE/TLS, or both TLS and SSL at the same time) on port 2481.

    Note: If you are using RelayServer, then the HTTPS listener is redundant an not required. The only deployment environment this option may be viable for would be BlackBerry/BES in a single server deployment.

    The values the administrator defines must be coordinated with the developer.
3. UltraLite/UltraLiteJ client dependency: E2EE is dependent on an UltraLite synchronization clients. Therefore, you can only use E2EE with UltraLite on Windows Mobile and Android devices. While BlackBerry clients can use UltraLiteJ clients, UltraLiteJ does not support the definition of HTTPS public certificate file paths. To use TLS (but not E2EE) with BlackBerry, you must install the trusted certificates on the device before you configure TLS.
4. Device configuration using templates: Administrators create an application template that contains initial configuration settings for the device application, including those required for E2EE. When a device client is registered, then onboarded, the encryption public key file and HTTPS public certificate file are provisioned wirelessly to the device. These artifacts can also be identified by the application developer as part of a synchronization profile.

See also

- *Provisioning Security Artifacts* on page 132

Changing Installed Certificates Used for Encryption

Unwired Server includes default certificates for all listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.

TLS/SSL/HTTPS all use default certificates that require changing. Different listeners require different tools.

- Use **keytool** to manage certificates for the encryption of DCN, OData, and DOE listeners. These listeners all use the key and truststores (`keysstore.jks`), because these listeners require mutual certificate authentication. OCSP is only used for these listeners.
- Use **creatcert** to manage certificates for replication encryption. OCSP is not supported for replication.

Irrespective of the tool used, you can follow these general steps.

1. Generate new production-ready certificates:
   - If you use a PKI system, ensure that the generated certificates and key pairs are signed by the certificate authority (CA) certificate that is widely trusted in your organization. Unwired Platform is compliant with certificates and key pairs generated from most well-known PKI systems. Sybase recommends that you use this option.
   - If you do not use a PKI system, use the **keytool** or **createcert** utility to generate new self-signed certificates.
2. Import production-ready certificates, then update the security profile to associate these files with the Unwired Server encrypted port.

   a) Use the appropriate tool to import the new production certificates into the primary Unwired Server keystore, if that listener requires it.
   b) Configure the listener properties.
   c) (Optional) If you are using a PKI system that includes OCSP and OCSP can be used by the listener, configure an OCSP responder. See *Enabling OCSP*.

See also

- *Certificate Creation (createcert) Utility* on page 183
- *Key Creation (createkey) Utility* on page 186

Modifying Default Synchronization Listener Properties with Production Values

Once you have determined the degree of secure communication you require, you may need to modify default synchronization listener property values to disable one or more ports or synchronization protocols.

Prerequisites

Ensure you have reviewed *Understanding Encyrption Requirements and Limitations*, and know what degree of secured or unsecured synchronization you require.

Task

For complete details on any of these properties, see *Configuring a Replication Listener* in the Sybase Control Center online help.

1. Open Sybase Control Center.
2. In the left navigation pane, expand the **Servers** folder and select the server to configure.
3. Select **Server Configuration** for the primary server.
4. In the right administration pane, click the **Replication** tab.
5. If you have added push synchronization to the application, select **Listener** from the menu bar.
6. Modify the protocol and port values:

    - To disable the HTTP port, unselect **Port**. Disabling this port means that you do not plan to use an unencrypted port, or use TLS for E2EE on this port (if you also disable all properties in step 9).

      Note: Do not unselect the HTTP port if you are using Relay Server. The RSOE cannot use the HTTPS port.
    - To change the default port value, delete port 2480 and enter a new value.
    - To disable the HTTPS , unselect **Secure port**. Disabling this port means that you do not plan to use HTTPS with SSL.
    - To change the default secure port value, delete port 2481 and enter a new value.

    Note: You cannot disable both ports.
7. Change the **Synchronization Cache Size** and **Thread Count** values if you need to make performance adjustements.
8. To change any default HTTPS with SSL properties (particularly to set new values for production-ready certificates for HTTPS), expand the **optional properties** section and modify these properties:

- Secure Sync Port Certificate – identifies the location of the security certificate used to encrypt and decrypt data transferred using SSL.
- Secure Sync Port Certificate Password – is used to decrypt the private certificate listed in the certificate file. You specify this password when you create the server certificate.
- Secure Sync Port Public Certificate – specify the file containing the public key that acts as the identity file for the synchronization port.
- Trusted Relay Server Certificate – if the Relay Server trusted certificate is configured, identifies the public security certificate location.

Note: If you have disabled the secure port, you do not need to configure these values.

9. To change any default E2EE properties (particularly to set new values for production ready certificates for E2EE), modify these properties:

- E2E Encryption Certificate Password – set the password to unlock the encryption certificate.
- E2E Encryption Certificate – specify the file containing the private key that acts as the identity file for Unwired Server.
- E2E Encryption Type – specify the asymmetric cipher used for key exchange for end-to-end encryption. You can only use RSA encryption.

Note: Leave E2EE values blank to disable end-to-end encryption.

Encryption Postrequisites

With the replication synchronization listener configured, ensure that the client application is provisioned with required artifacts and has connections configured accordingly.

Encrypting Other Listeners for Unwired Server

By default all other Unwired Platfrom listeners are encrypted using SSL. However, if you need to modify this configuration, review these steps.

1. *Changing Installed Certificates Used for Encryption*

   Unwired Server includes default certificates for all listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.
2. *Changing Keystore and Truststore Passwords*

   The Unwired Platform (used by both Unwired Server and Sybase Control Center to manage certificates and keys) keystore and truststore locations are protected by a password. In production environments, replacing default passwords is encouraged.
3. *Defining Certificates for SSL Encryption*

For the primary server, specify keystore and truststore certificates to be used for SSL encryption of Unwired Platform communication ports. All security profiles use the same keystore and truststore.

4. *Creating an SSL Security Profile in Sybase Control Center*

   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.
5. *Enabling OCSP*

   (Optional) Enable OCSP (Online Certificate Status Protocol) to determine the status of a certificate used to authenticate a subject: current, expired, or unknown. OCSP configuration is enabled as part of server level SSL configuration. OCSP checking must be enabled if you are using the CertificateAuthenticationLoginModule and have set Enable revocation checking to true.

See also

- *Encrypting Synchronization for Replication Payloads* on page 102

Changing Keystore and Truststore Passwords

The Unwired Platform (used by both Unwired Server and Sybase Control Center to manage certificates and keys) keystore and truststore locations are protected by a password. In production environments, replacing default passwords is encouraged.

Prerequisites

Before you begin, back up the contents of `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\Repository`.

Task

In production environments, use the keytool utility to change the default passwords for the keystore and truststore locations.

1. Open a command prompt window from this location: `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\Repository\Security`.
2. Run commands to change the current password for the keystore, truststore, and private key entries as required for your environment.

   You must enter the same password for a keystore and each of the private entries associated with that store.

   There is no provision in Sybase Control Center to specify a different password for the private key aliases.

For the keystore password, use: `keytool -storepasswd -new` *`NewPwd`* `-keystore Security\keystore.jks`

For the truststore password, use: `keytool -storepasswd -new` *`NewPwd`* `-truststore Security\truststore.jks`

For private key entries in keystore, use: `keytool -keypasswd -alias` *`Name`* `-new` *`NewPwd`* `-keystore Security\keystore.jks`

3. At the prompt, enter the current password.

   If this is the first time changing the password, enter the default password of `changeit`. Otherwise, enter the current password.
4. In Sybase Control Center, configure the Primary Unwired Server SSL certificates to use these passwords. If these certificates are already configured, update the passwords currently configured.

   Click **Servers >** ***PrimaryServerName*** **> Server Configuration > General**, then click the **SSL Configuration** tab. For details, see *Defining Certificates for SSL Encryption* .

   If you do not ensure the correct password is set, you can expect a connection failure. See *Keystore Tampering Message Suggests that Connection with Unwired Server Fails* in the *Troubleshooting* guide.
5. Restart all Unwired Platform services using the Windows Control Panel services tool.

See also

- *Changing Installed Certificates Used for Encryption* on page 105

Defining Certificates for SSL Encryption

For the primary server, specify keystore and truststore certificates to be used for SSL encryption of Unwired Platform communication ports. All security profiles use the same keystore and truststore.

For secondary Unwired Servers, SSL properties are synchronized from the primary server. Therefore for secondary servers, these properties are still visible, but cannot be edited.

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. To configure SSL encryption for all security profiles, complete these fields:
   - **Keystore Location –** the full path name indicating the location where the keys and certificates are stored. Certificates used for administration and data change notification ports are stored in the keystore. The path should be relative to *`<Unwired Platform_InstallDir>`*`\`*`UnwiredPlatform-XX`*`\Servers\UnwiredServer`.

- **Keystore Password –** the password that secures the key store.
- **Truststore Location –** the full path name for the public key certificate storage file. The Certificate Authority (CA) certificates used to sign certificates store their public keys in the truststore. The path should be relative to `<Unwired Platform_InstallDir>\UnwiredPlatform-XX\Servers \UnwiredServer`.
- **Truststore Password –** the password that secures the truststore.

Note: If at any point you have changed the password for the keystore and truststore with keytool, then you must remember to update the password here as well.

6. Click **Save**.

Next

Create an SSL security profile that uses the selected certificates.

Creating an SSL Security Profile in Sybase Control Center

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.

Note: A security profile can be used by one or more servers in a cluster, but cannot used by multiple clusters.

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. 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:

      If the security profile authenticates only the server, then only the server must provide a certificate to be accepted or rejected by the client. If the security profile authenticates both the client and the server, then 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.

Profile	Authenticates	Cipher suites
intl	server	• SA_EX-PORT_WITH_RC4_40_MD5 • RSA_EX-PORT_WITH_DES40_CBC_SHA
intl_mutual	client/server	• RSA_EX-PORT_WITH_RC4_40_MD5 • RSA_EX-PORT_WITH_DES40_CBC_SHA
strong	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
strong_mutual	client/server For example, this is the required option for mutual authentication of Unwired Platform and Gateway.	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
domestic	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EX-PORT_WITH_RC4_40_MD5 • RSA_EX-PORT_WITH_DES40_CBC_SHA • TLS_RSA_WITH_NULL_MD5 • TLS_RSA_WITH_NULL_SHA

Profile	Authenticates	Cipher suites
domestic_mutual	client/server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • RSA_WITH_NULL_MD5 • RSA_WITH_NULL_SHA

6. Click **Save**.
7. From the **Communication Ports** menu, assign the security profile to the desired management or communication ports.

Next

If you configure a secure port on one server, you must enable it on every node in the cluster, then restart all servers in the cluster to commit the configuration changes.

Enabling OCSP

(Optional) Enable OCSP (Online Certificate Status Protocol) to determine the status of a certificate used to authenticate a subject: current, expired, or unknown. OCSP configuration is enabled as part of server level SSL configuration. OCSP checking must be enabled if you are using the CertificateAuthenticationLoginModule and have set Enable revocation checking to true.

Enable OCSP for an Unwired Server when configuring SSL.

1. To enable OCSP when doing certificate revocation checking, check **Enable OCSP**.
2. Configure the responder properties (location and certificate information):

Responder Property	Details
URL	A URL to responder, including its port. For example, `https://ocsp.example.net:80`.

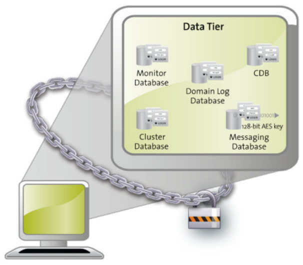
Responder Property	Details
Certificate subject name	The subject name of the responder's certificate. By default, the certificate of the OCSP responder is that of the issuer of the certificate being validated. Its value is a string distinguished name (defined in RFC 2253), which identifies a certificate in the set of certificates supplied during cert path validation. If the subject name alone is not sufficient to uniquely identify the certificate, the subject value and serial number properties must be used instead. When the certificate subject name is set, the certificate issuer name and certificate serial number are ignored. For example, `CN=MyEnterprise, O=XYZCorp`.
Certificate issuer name	The issuer name of the responder certificate. For example, `CN=OCSP Responder, O=XYZCorp`.
Certificate serial number	The serial number of the responder certificate.

See also

- *Creating an SSL Security Profile in Sybase Control Center* on page 110

CHAPTER 5 Data Tier Security

The data tier consists of multiple databases, each of which plays a unique role in Unwired Platform, and contains various types of sensitive data that must be secured.



Securing the Data Infrastructure

Secure data by first protecting the infrastructure on which it resides, then securing runtime databases.

1. *Setting File System Permissions*

   Unwired Platform runs as a collection of Windows services. During installation, you are prompted with a Logon as request. The credentials collected are then used to run the service under that account. In a cluster installation, the same Windows user would be configured for all installations and respective Window services that are subsequently installed.
2. *Securing Backup Artifacts*

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

Setting File System Permissions

Unwired Platform runs as a collection of Windows services. During installation, you are prompted with a **Logon as** request. The credentials collected are then used to run the service under that account. In a cluster installation, the same Windows user would be configured for all installations and respective Window services that are subsequently installed.

You can restrict permissions after installation by removing most users and groups from the Unwired Platform installation directory.

1. Open File Explorer.

2. Right-click `<UnwiredPlatform_InstallDir>`, and click **Properties**.
3. On the **Security** tab, click **Advanced**.
4. Unselect **Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.**.
5. In the confirmation pop-up, choose **Copy**, then select **Replace permission entries on all child objects with entries shown here that apply to child objects.**.
6. In the table of Permission entries, remove all users except the user account that was configured as the Logon user for the Windows services. If another user is responsible for some activities extend the necessary permissions to this administrator. For example, if the individual is only reading log files, you may choose to limit permissions to read only.
7. Click **OK**.

Securing Backup Artifacts

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

1. Follow the instructions in *System Administration* for using **dbvalid** and **dbbackup** to back up the platform runtime data.

   Sybase recommends that you encrypt your database and all backups.
2. Protect these backups with Administrator and SYSTEM permissions.
3. Perform any additional enterprise security requirements.

Securing Data Tier Databases

Secure all databases installed as the Unwired Platform data tier. You can change DBA passwords, grant DBA permissions to other users, and encrypt data and logs.

See also

- *Securing the Data Infrastructure* on page 115
- *Encrypting Device Data* on page 12

Changing DBA Passwords for SQLAnywhere Databases in a Cluster Deployment

By default, Unwired Platform uses multiple SQLAnywhere databases to support the server runtime environment and transactions. Unwired Server accesses these databases with the DBA user identity.

During installation, enter a custom password for the DBA user on each of the SQLAnywhere databases used by the runtime: the cache database (CDB), the cluster database, the log database, and monitor database.

In a cluster deployment, these servers are used:

- The monitor and domain log databases use `LogDataDB`.
- The cache database uses `CacheDB`.
- The cluster database uses `ClusterDB`.

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

   For a list of database services, search for *Unwired Platform Windows Services* in *System Administration*.
2. On data tier host:
   a) Set the location of the BIN directory for your operating system (32-bit or 64-bit):
   ```
   set SQLANY12_BIN=<UnwiredPlatform_InstallDir>\Servers
   \SQLAnywhere12\bin<32|64>
   ```
   b) Set the path to the data:

   If you are using a single node, run:
   ```
   set DATA_PATH= <UnwiredPlatform_InstallDir>\Servers
   \UnwiredServer\data
   ```
   If you are using a cluster deployment, run:
   ```
   set DATA_PATH= <UnwiredPlatform_InstallDir>\data\CDB
   ```
3. Use **dbisql** to change passwords for each database as required:

   For the cache database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=default;DBF=%DATA_PATH%
   \default.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the cluster database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=clusterdb;DBF=%DATA_PATH%
   \clusterdb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the monitor database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=monitordb;DBF=%DATA_PATH%
   \monitordb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the domain log database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=domainlogdb;DBF=%DATA_PATH
   %\domainlogdb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
4. To register the change with the runtime, run this command on each Unwired Server node host:
   ```
   Register-dsn.bat cdb.install_type cdb.serverhost cdb.serverport
   cdb.username
    %CDB_PASSWORD% cdb.servername cldb.dsnname %CLDB_PASSWORD%
   %MONITORDB_PASSWORD% %DOMAINLOGDB_PASSWORD%
   ```
   To see the values used in the properties of this command, open the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer`

`\Repository\Instance\com\sybase\sup\server\SUPServer\sup.properties` file and search for the corresponding property.

5. If you receive an `Invalid user ID` or `Invalid password` error, you may have already changed the password from "sql" to different one). In this case:
   a) Backup **register-dsn.bat**.
   b) Open this file in a text editor and locate:

   ```
   IF "default" == "%CDB_INSTALLTYPE%" (
       echo Changing DBA password for databases ...
       "%SQLANY12_BIN%\dbisqlc" -q -c "Server=default;DBF=
   %DJC_HOME%\data\default.db;UID=DBA;PWD=sql" grant connect to
   dba identified by %CDB_PASSWORD%
       "%SQLANY12_BIN%\dbisqlc" -q -c "Server=clusterdb;DBF=
   %DJC_HOME%\data\clusterdb.db;UID=DBA;PWD=sql" grant connect to
   dba identified by %CLDB_PASSWORD%
       "%SQLANY12_BIN%\dbisqlc" -q -c "Server=monitordb;DBF=
   %DJC_HOME%\data\monitordb.db;UID=DBA;PWD=sql" grant connect to
   dba identified by %MONITORDB_PASSWORD%
       "%SQLANY12_BIN%\dbisqlc" -q -c "Server=domainlogdb;DBF=
   %DJC_HOME%\data\domainlogdb.db;UID=DBA;PWD=sql" grant connect
   to dba identified by %DOMAINLOGDB_PASSWORD%
   )
   ```

   c) Replace `PWD=sql` with the `PWD=`*`PreviousPassword`*.
6. To register the change with the synchronization server, run:

   ```
   run-ant-config.bat configure-mlsrv.ini configure-sup -
   Dsqlany12.bin=%SQLANY12_BIN%
   ```

   where *%SQLANY12_BIN%* is substituted with the path value recorded in the `asa-setenv.bat`.
7. Restart all database services, then all Unwired Servers.

Changing DBA Passwords for SQLAnywhere Databases in a Single-Node Installation

By default, Unwired Platform uses multiple SQLAnywhere databases to support the server runtime environment and transactions. Unwired Server accesses these databases with the DBA user identity.

During installation, enter a custom password for the DBA user on each of the SQLAnywhere databases used by the runtime: the cache database (CDB), the cluster database, the log database, and monitor database.

In single-node deployment, a single database server named `CacheDB` supports all installed databases.

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

   For a list of database services, search for *Unwired Platform Windows Services* in *System Administration*.

2. On data tier host:
   a) Set the location of the BIN directory for your operating system (32-bit or 64-bit):
   ```
   set SQLANY12_BIN=<UnwiredPlatform_InstallDir>\Servers
   \SQLAnywhere12\bin<32|64>
   ```
   b) Set the path to the data:

   If you are using a single node, run:
   ```
   set DATA_PATH= <UnwiredPlatform_InstallDir>\Servers
   \UnwiredServer\data
   ```
   If you are using a cluster deployment, run:
   ```
   set DATA_PATH= <UnwiredPlatform_InstallDir>\data\CDB
   ```
3. Use **dbisql** to change passwords for each database as required:

   For the cache database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=default;DBF=%DATA_PATH%
   \default.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the cluster database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=clusterdb;DBF=%DATA_PATH%
   \clusterdb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the monitor database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=monitordb;DBF=%DATA_PATH%
   \monitordb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the domain log database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=domainlogdb;DBF=%DATA_PATH
   %\domainlogdb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
4. If you receive an `Invalid user ID` or `Invalid password`:
   a) Backup **register-dsn.bat**.
   b) Open this file in a text editor and locate:
   ```
   IF "default" == "%CDB_INSTALLTYPE%" (
       echo Changing DBA password for databases ...
       "%SQLANY12_BIN%\dbisqlc" -q -c "Server=default;DBF=
   %DJC_HOME%\data\default.db;UID=DBA;PWD=sql" grant connect to
   dba identified by %CDB_PASSWORD%
       "%SQLANY12_BIN%\dbisqlc" -q -c "Server=clusterdb;DBF=
   %DJC_HOME%\data\clusterdb.db;UID=DBA;PWD=sql" grant connect to
   dba identified by %CLDB_PASSWORD%
       "%SQLANY12_BIN%\dbisqlc" -q -c "Server=monitordb;DBF=
   %DJC_HOME%\data\monitordb.db;UID=DBA;PWD=sql" grant connect to
   dba identified by %MONITORDB_PASSWORD%
       "%SQLANY12_BIN%\dbisqlc" -q -c "Server=domainlogdb;DBF=
   %DJC_HOME%\data\domainlogdb.db;UID=DBA;PWD=sql" grant connect
   ```

```
to dba identified by %DOMAINLOGDB_PASSWORD%
)
```

c) Replace `PWD=sql` with the `PWD=`*`NewPassword`*.

5. To register the change with the runtime, run this command:

```
Register-dsn.bat cdb.install_type cdb.serverhost cdb.serverport
cdb.username
 %CDB_PASSWORD% cdb.servername cldb.dsnname %CLDB_PASSWORD%
%MONITORDB_PASSWORD% %DOMAINLOGDB_PASSWORD%
```

To see the values used in the properties of this command, open the `<`*`UnwiredPlatform_InstallDir`*`>\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer\sup.properties` file and search for the corresponding property.

6. To register the change with the synchronization server, run:

```
run-ant-config.bat configure-mlsrv.ini configure-sup -
Dsqlany12.bin=%SQLANY12_BIN%
```

where *%SQLANY12_BIN%* is substituted with the path value recorded in the `asa-setenv.bat`.

7. Restart all database services, then all Unwired Servers.

Encrypting Data and Log Outputs

Sybase SQL Anywhere database files and log files that are used as part of the Unwired Platform data tier can be encrypted. The databases that use this database type are the CDB, the monitoring database, and the domain log database.

1. Stop all Sybase Unwired Platform services.

2. Launch dbisql from *<UnwiredPlatform_InstallDir>*\Servers\SQLAnywhere*XX*\BIN*XX*.

3. Connect to a database, other than the client database you want to encrypt.

4. From **dbisql**, issue:

```
CREATE ENCRYPTED DATABASE ‘newdbfile’ FROM ‘existingdbfile’ KEY
‘someKey’ ALGORITHM ‘algorithm’
```

Supported algorithms include:

- SIMPLE
- AES
- AES256
- AES_FIPS
- AES256_FIPS

Note: FIPS options are available only as a separately licensed option for SQLAnywhere.

5. Once the database files and log files are encrypted:

a) Shut down the database server.

   b) Restart the database server with the -ek <encryption key> database option.

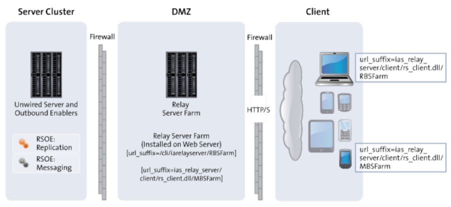
      This modifies the server startup to use the encrypted copy of the database file.

6. Restart all stopped services.

CHAPTER 6 DMZ Security

DMZ security involves controlling Internet traffic to private networks by installing a Relay Server between your inner and outer firewalls.

The outer firewall has HTTP and HTTPS ports open to allow client Internet traffic to reach the Relay Server.


Server Cluster
DMZ
Client
Firewall
Firewall
Unwired Server and Outbound Enablers
RSOE: Replication
RSOE: Messaging
Relay Server Farm
Relay Server Farm (Installed on Web Server)
[url_suffix=/cli/iarelayserver/RBSFarm]
[url_suffix=ias_relay_server/client/rs_client.dll/MBSFarm]
HTTP/S
url_suffix=ias_relay_server/client/rs_client.dll/RBSFarm
url_suffix=ias_relay_server/client/rs_client.dll/MBSFarm

Relay Server as Firewall Protection

The Relay Server is a pair of Web server plug-ins, which you can install on an Internet Information Service (IIS) server on Windows, or on the Apache Web server on Linux.

The Relay Server is intended to run between a company's inner and outer firewalls. The outer firewall has HTTP and HTTPS ports open to allow client Internet traffic to reach the Relay Server. The client's URL includes the address of the client-side plug-in of the Relay Server and the name of the back-end Sybase Unwired Platform "farm" the client is trying to reach. A farm includes multiple Relay Servers for load balancing and fault tolerance. The network administrator must install a load balancer in front of the Relay Servers. The load balancer is not included with Sybase Unwired Platform. To make the interaction secure, clients should use end-to-end encryption.

The server-side plug-in accepts connections from various Relay Server Outbound Enabler (RSOE) processes, which indicate to the Relay Server what back-end farm each process represents. The Relay Server matches the farm name in the client's request to a server-side plug-in connection, and routes the client's request contents to that connection. Other than the farm name in the request URL, the Relay Server knows nothing about the content of these messages. The clients are not authenticated or authorized in any way. The client information is in memory and therefore is not susceptible to interception or modification. But, if the administrator turns certain tracing options up very high, data may get copied to log files. If end-to-end encryption is used, the data is undecipherable.

Security administrators secure the Relay Server as they would with any other Web server or proxy server they run between firewalls, so the same security precautions should be taken of setting up a proxy server.

See also

- *RSOE as the Unwired Server Protection* on page 124
- *Relay Server and RSOE Communication Security* on page 124
- *Configuring Connection Properties for Relay Server Components* on page 125

RSOE as the Unwired Server Protection

One RSOE process is installed in each Unwired Server cluster member, in front of each synchronization subcomponent that communicates with a client. Replication service components and messaging service components both use RSOEs attached to their communication ports.

The RSOE configuration enables the Relay Server to identify the RSOE and connect to it. The RSOE configuration also has a single port number that enables the RSOE to make an `http://localhost:port` connection whenever a client request comes to it from the Relay Server.

See also

- *Relay Server as Firewall Protection* on page 123
- *Relay Server and RSOE Communication Security* on page 124
- *Configuring Connection Properties for Relay Server Components* on page 125

Relay Server and RSOE Communication Security

The RSOE runs on the same computer as an Unwired Server and is configured with the address of a Relay Server (the inner firewall is open to outgoing traffic, but not incoming traffic).

The RSOE connects to the Relay Server via HTTP or HTTPS and identifies itself through the Media Access Control (MAC) address, security token, and the back-end Sybase Unwired Platform farm it services. The Relay Server identifies the RSOE's authenticity. If Relay Server accepts the RSOE's identity, it sends RSOE a list of all other RSOEsin the Relay Server farm. The RSOE establishes a blocking GET HTTP request to eachfarm member. When a Relay Server receives a client request for a given Sybase Unwired Platform farm, it picks one of the available RSOE connections and sends the client request there.

In this way, the network administrator need not open inner firewall ports to allow connection requests into the intranet. All connection requests come from within the intranet. Avoiding firewall portholes protects the intranet from hackers who breach the outer firewall.

This network traffic contains exactly the same content, and thus the same security concerns as network communication between the device application or database and the Relay Server.

See also

- *Relay Server as Firewall Protection* on page 123
- *RSOE as the Unwired Server Protection* on page 124
- *Configuring Connection Properties for Relay Server Components* on page 125

Configuring Connection Properties for Relay Server Components

In most highly available deployments, you configure both Relay Server and RSOE to use HTTP when connecting to Unwired Server on the corporate LAN. In more specialized, less available deployments (for example, where BES is inside the corporate LAN and is configured to connect directly to Unwired Server without any load-balancing by Relay Server), use HTTPS.

Configure both Relay Server and Outbound Enablerconnections:

1. *Configuring Relay Server Connection Properties*

   Configure the connection type used by Relay Server to connect to the Unwired Server.
2. *Configuring Outbound Enabler Connection Properties*

   Outbound Enabler establishes two connections. You can configure connections from the Outbound Enabler to Relay Server to use either HTTP or HTTPS. However, connections from the Outbound Enabler to Unwired Server can only use HTTP, so this connection does not require configuration.

See also

- *Relay Server as Firewall Protection* on page 123
- *RSOE as the Unwired Server Protection* on page 124
- *Relay Server and RSOE Communication Security* on page 124

Configuring Relay Server Connection Properties

Configure the connection type used by Relay Server to connect to the Unwired Server.

Prerequisites

If you are using a load balancer, configure it with the same properties as Relay Server.

Task

1. In the navigation pane, click the Unwired Server cluster name.

2. In the administration pane, click the **Relay Servers** tab.
3. Click **New**.
4. When you reach the **General** properties page, configure these properties:
   - In highly available deployments where Relay Server is deployed to the DMZ, enable the HTTP port.
   - When Relay Server is installed on the corporate LAN, enable the HTTPS port.
5. Configure all remaining properties as documented in *Creating a Custom Relay Server Configuration* of the *Installation Guide for Runtime*.

Configuring Outbound Enabler Connection Properties

Outbound Enabler establishes two connections. You can configure connections from the Outbound Enabler to Relay Server to use either HTTP or HTTPS. However, connections from the Outbound Enabler to Unwired Server can only use HTTP, so this connection does not require configuration.

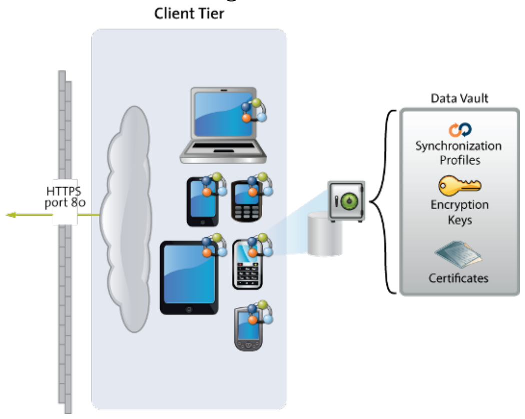
After you configure Relay Server, configure Outbound Enablers on each Unwired Server node.

1. In the navigation pane, click **Servers > <*ServerNode*> > Server Configuration**.
2. In the administration pane, select the **Outbound Enabler** tab, then click **New**.
3. In the General properties page, ensure you configure the Relay Server port for the type of connection you require (either HTTP or HTTPS).
4. If you choose HTTPS for the Outbound Enabler's client connection, either import the Relay Server certificate or that of the RelayServer's certificate signing CA into the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\Security` directory on the primary Unwired Server.
5. Configure the Outbound Enabler's Certificate file, and optionally the Trusted certificate (when the certificate file contains multiple certificates) with appropriate values.
6. Configure all other values as described in *Configuring the Outbound Enablers* of the *Installation Guide for Runtime*.

CHAPTER 7 Device Security

You can combine multiple mechanisms to fully secure devices. In addition to using the built-in security features of both the device or Unwired Platform, Sybase recommends that you also use Afaria so you can remotely initiate security features as required.

Unwired Platform security features for devices include data encryption, login screens, and data vaults for storing sensitive data.



Limiting Application Access

Application access to Unwired Platform runtime is tightly controlled: before a user can access a mobile application, he or she must provide a passcode; before the application can access the runtime, the application must be registered and provisioned with required connections and security configurations.

1. *Encrypting Device Data*
   Encrypting all data on the device client requires multiple techniques.
2. *Registering Applications, Devices, and Users*
   Before messaging, workflow, or OData applications can access the runtime, the user, device, and application must be identified by first registering with Unwired Server and pairing them with a device and user entry. Only when all three entities are known, can subscriptions can be made or data synchronized. In contrast, replication users must know only their authentication credentials (passed on to backend security stores via the security provider).
3. *Locking and Unlocking a Subscription*
   (Not applicable to Online Data Proxy) Create a subscription template to lock or unlock a subscription. A subscription determines what data set the device user receives upon

synchronization and how frequently the synchronization can occur. Lock the subscription to prevent modification to the template and control the synchronization frequency

4. *Locking and Unlocking Application Connections*

Lock or unlock connections to control which users are allowed to synchronize data. Locking an application connection is an effective way to disable a specific user without making changes to the security profile configuration to which he or she belongs.

Encrypting Device Data

Encrypting all data on the device client requires multiple techniques.

Some Unwired Platform components do not support encryption. Review this table to see which components can enable this security feature.

Component	Implementation notes
Device data	Sybase recommend full device encryption with Afaria. See the Afaria documentation for details.
Device client database	(Not applicable to Online Data Proxy) A `<package>DB.generateEncryptionKey()` method in the Object API for MBO packages should always be used during application initialization. It computes a random AES-256 bit encryption key used to encrypt the client database. The encryption key is stored in the data vault.
Data vault	The DataVault APIs provide a secure way to persist and encrypt data on the device. The data vault uses AES-256 symmetric encryption of all its contents. The AES key is computed as a hash of the passcode provided and a "salt" value that can be supplied by the device application developer, or automatically generated through the API.

See also

- *Securing Data Tier Databases* on page 116
- *Registering Applications, Devices, and Users* on page 128

Registering Applications, Devices, and Users

Before messaging, workflow, or OData applications can access the runtime, the user, device, and application must be identified by first registering with Unwired Server and pairing them with a device and user entry. Only when all three entities are known, can subscriptions can be made or data synchronized. In contrast, replication users must know only their authentication credentials (passed on to backend security stores via the security provider).

In Sybase Control Center, Platform administrators set up an application connection template for a messaging, workflow, or OData application, wherein they can specify whether or not to Enable Automatic Registration.

- When automatic registration is enabled, a device user need only provide valid Unwired Platform credentials that are defined as part of the security configuration.
- When automatic registration is disabled, the platform administrator must provide the user a username/passcode out-of-band. This is the passcode initially required by login screens to access the application for the first time, and expires within a predetermined time period (72 hours, by default).

See also

- *Locking and Unlocking a Subscription* on page 129

Registering Application Connections

Devices can be registered using either a one-time passcode during the registration of the device or application, or to allow automatic registration.

When a package is deployed, an Application Connection Template is automatically created for it. As long as the user is able to authenticate to the security configuration associated with the application, they are registered. If automatic registration is disabled then the administrator must generate a passcode. For details, see *Sybase Control Center* online help and search for *Registration and Onboarding*.

1. Select the application template, and click the **Properties** button.
2. Navigate to the **Application Settings** tab and set the **Automatic Registration Enabled** property to True or False. If True, no one-time passcode is required.

Manually Creating Applications

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

Locking and Unlocking a Subscription

(Not applicable to Online Data Proxy) Create a subscription template to lock or unlock a subscription. A subscription determines what data set the device user receives upon synchronization and how frequently the synchronization can occur. Lock the subscription to prevent modification to the template and control the synchronization frequency

1. In the left navigation pane, expand the **Packages** folder and select the replication-based package to configure.
2. In the right administration pane, click the **Subscriptions** tab.
3. From the menu bar, select **Templates**, then click **New**.
4. Select **Admin Lock** to prevent device users from modifying the push synchronization state or sync interval value configured in the subscription. If the admin lock is disabled, the

device client user can change these properties, and these changes take effect the next time the client user synchronizes the package to which the subscription applies.

See also

- *Registering Applications, Devices, and Users* on page 128

Locking and Unlocking Application Connections

Lock or unlock connections to control which users are allowed to synchronize data. Locking an application connection is an effective way to disable a specific user without making changes to the security profile configuration to which he or she belongs.

1. In the left navigation pane, select the**Applications** node.
2. In the right administration pane, select the **Application Connections** tab.
3. Select the application connection you want to manage, and:
   - If the connection is currently unlocked and you want to disable synchronization, click **Lock**.
   - If the connection is currently locked and you want to enable synchronization, click **Unlock**.
4. In the confirmation dialog, click **OK**.

Securing Sensitive Data On-Device with Data Vault

(Not applicable to Hybrid Workflow Container) Developers should use a data vault with device applications to securely store “secrets” on the device. Data vaults are added using the DataVault API.

The data vault holds sensitive artifacts securely, because all data or artifacts in the data vault is strongly encrypted with an AES-256 bit key. Contents can include encryption keys, user and application login credentials, sync profile settings, certificates (as BLOBS).

The data vault requires a password to unlock and access the data from the application. Therefore, a device application must prompt the user to enter this password when the application is opened. Once unlocked, the application can retrieve any other secrets from the vault as needed, all without prompting the user.

Administrators can define a password policy through Sybase Control Center that defines the requirements for an acceptable password. The password policy is stored in the server-side settings database and the client gets those settings when it connects to Unwired Server as part of the settings exchange protocol.

When the client receives the password policy settings, it can populate the settings objects to the data vault. The datavault stores the settings. The client uses the DataVault API to create a vault with a defaut password, set the password policy, and change the password to a password compatible with the policy. If the password is not changed after setting a passwport policy, the application will throw an exception if you then try to access the application or unlock the vault with an incompatible password.

Administrators should discuss the data vault strategy before it is implemented, especially regarding:

- **Failed logins –** Developers can set the number of failed login attempts allowed before the data vault is deleted. Once the vault is deleted the encrypted databases will be un-useable. The application will need to be re-installed, or re-initialized from scratch including deleting the database files to recover
- **Timeouts –** Developers can also set a timeout value so that the data vault locks itself when it's not in use. The user must re-enter the vault password to resume using the application.

For more details about the data vault, see *DataVault* in the developer guide for your application type.

See also

- *Using Login Screens for Data Vaults* on page 131

Using Login Screens for Data Vaults

An application that implements a login screen is considered to be secure. Mobile application developers are responsible for creating login screens for the applications they create. A login screen allows the device user to enter a passcode to unlock the data vault.

A secure application that uses a login screen:

1. Prompts the user to enter the datavault passcode to open the application and get access to the local client database. If the wrong passcode is used, the application is rendered useless: the key that encrypts and decrypts data in the vault cannot be used to access data until this code is accurately entered.

   After a certain amount of time passes, the login in screen can be redeployed to prompt the user to re-enter the passcode.
2. Can be locked out after a configured number of retries.
3. Can self-destruct after a set number of incorrect passcode attempts.

   When this occurs, the device user must uninstall, reinstall, then perform an initial synchronization to recover from a destroyed data vault.

To implement a login screen you must create the login and the define the password. The screen and the password unlock the DataVault. Unlocking the vault enables access to application data off-line or on-line. In contrast, Workflow applications can require user credentials that must be checked against Unwired Server on-line before granting access to Workflow content.

The password is initially defined when you configure the property values required to enable an authenticated HTTPS connection. However, you can allow users to change this password. For information about password definition see *changePassword* in the Developer guide for your application type.

See also

- *Securing Sensitive Data On-Device with Data Vault* on page 13

Provisioning Security Artifacts

Typically, you must preprovision Unwired Platform security artifacts before applications can be used. The manner by which an artifact is provisioned depends on the artifacts themselves, the device types used, and your deployment environment.

See also

- *Modifying Default Synchronization Listener Properties with Production Values* on page 106
- *Establishing Encrypted Application Connections* on page 137
- *Unwired Server and Device Application Communications* on page 5
- *Encrypting Synchronization for Replication Payloads* on page 102
- *Understanding Encryption Requirements and Limitations* on page 104

Security Artifacts That Require Provisioning

Certain artifacts must be provisioned to the device before the device can connect to the runtime.

- **SSO certificates –** Certificates that identify the user for SSO-enabled applications for SAP backends.
- **Encryption keys –** Keys that encrypt the connection to Relay Server
- **Security profiles –** Profiles saved to the device and contain sensitive information.

- **User and application login credentials –** User names and password that allow a user to access backend data.
- **Connection properties –** The values for server, host, port, Relay Server farm and other property values that can all be provisioned by Afaria. The device user has can also manually enter these values upon initial connection, and these values are stored in the data vault.

See also

- *Provisioning Methods by Application Type* on page 133
- *Provisioning With Afaria: Security Considerations* on page 135
- *Provisioning with Unwired Server* on page 136

Provisioning Methods by Application Type

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

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

Device Application	Method	Details
BlackBerry native – once applications provisioned and installed, the application appears in Downloads. However, device users can move it to a different location. If device users reinstall the application from a link or URL, or using Desktop Manager, the BlackBerry device remembers the installation location.	BlackBerry Enterprise Server (BES) push notifications	When the BlackBerry device activates, it automatically pairs with the BES and downloads the application. See *http://www.blackberry.com/btsc/search.do?cmd=displayKC&docType=kc&externalId=KB03748* for step-by-step instructions. Also see *BlackBerry Provisioning with BES* in *System Administration*.
	URL/link to installation files	The administrator stages the OTA files in a Web-accessible location and notifies BlackBerry device users via an e-mail message with a link to the JAD file.
iOS native – users of iPhones tend to self-manage their devices: device users download application files to their iOS devices and synchronize updates as required. As Administrator, you must ensure users are notified of required downloads.	Apple Push Notification Service (APNS)	APNS allows users to receive notifications on iPhones. Each application that supports APNS must be listed in Sybase Control Center with its certificate and application name. See *Provisioning with iOS APNS* in *System Administration*.

Device Application	Method	Details
Messaging native applications – may require direct connections either on the LAN or WIFI. This connection prevents man-in-the-middle attacks that might otherwise occur.	Direct Unwired Server connections	If you are not using Afaria for messaging clients, you must first install the client application then connect to corporate LAN using WIFI or any other method of your choosing in order to provision devices with required files. See *Provisioning with Unwired Server* in *System Administration*.
Hybrid Web Container – because this native application embeds a browser control supplied by the device OS, these devices require specific provisioning techniques.	Configuration Files	The Hybrid Web Container supports only the provisioning from a file method. Provisioning of the Hybrid Web Container provisions the container itself (and therefore all the apps in the container). See *Provisioning with Configuration Files* in *System Administration*.
Heterogeneous applications (including online data proxy and DOEC) –if you are supporting a heterogeneous device environment, these methods are the best way to provision all devices using the same method. Which method you choose depends on whether or not you want devices connecting directly to Unwired Server.	Afaria and Configuration Files	Automate the process of provisioning applications by enabling a group of applications to be initially provisioned with a set of parameters from a configuration file using Afaria. See *Provisioning with Afaria* in *System Administration*.
Windows, Windows Mobile, and BlackBerry	Desktop Management and cradle	Installs the native application when the device is synchronized via a computer. See your device documentation for details.

See also

- *Security Artifacts That Require Provisioning* on page 132
- *Provisioning With Afaria: Security Considerations* on page 135
- *Provisioning with Unwired Server* on page 136

Provisioning With Afaria: Security Considerations

Applications must be provisioned with name=value pair parameters to connect to Unwired Server. Those parameters can include connection parameters for Unwired Server including the server's public key, and an X.509 certificate for authentication through SSO. Administrators can automate the process of provisioning applications by enabling a group of applications to be initially provisioned with a set of parameters from a configuration file using Afaria. Developers include API calls in the application to retrieve the provisioning data and certificate.

This task assumes that an Afaria client is co-installed on the device, that the user has enrolled in Afaria management, and that a Certificate Authority server is available to be configured for use with an Afaria portal package.

1. Ensure that required configuration files, keys (for E2EE), or other artifacts are available.

   The administrator creates an initial configuration file for applications which are to be deployed and provisioned through Afaria. See *System Administration* for the format of this file, and instructions on how to get the public key of the Unwired Server, for inclusion in the file.).

   Note: Without Afaria, the initial configuration file can be manually placed on a device, and consumed by the application. iOS devices do not support this capability because the application sandbox cannot be manually accessed. See *System Administration* to learn what the name of the file must be, and where it needs to be placed (it varies by platform).

2. Ensure the mobile application developer includes code in the application to allow the application client to make request for a certificate and provisioning data to Afaria: validate that this key exists on the device and is properly provisioned.

   a) Include code to check whether the application is provisioned, and if not, retrieve a provisioning file using Afaria. The provisioning file includes the public key of the Unwired Server.

   b) If an application wishes to use certificate-based authentication, the application developer includes code in the application to retrieve an X.509 certificate using Afaria and a Certificate Authority configured for the Afaria portal package. The application consumes, uses, or deletes the certificate as required. The application developer sets up a synchronization profile, and presents the user's certificate for authentication.

      The application developer also includes in the synchronization profile any other required application configuration parameters, such as encryption keys used for synchronization.

      The developer also includes code to prompt the application user to enter the Common Name and Challenge Code.

   For details on client APIs, see the *Developer Guide* for your device type.

3. The Afaria administrator performs the following tasks to creates a portal package to serve deployments for a group of application of a specified device platform on an Unwired Server:
   a) Defines the Certificate Authority server for the portal package.
   b) Includes the portal package into a Group Profile.
   c) Imports the configuration file to provision applications in that portal package.
   d) Enables Simple Certificate Enrollment Protocol (SCEP) in Afaria Server.

      Use SCEP to:
      - Create or obtain a user certificate on behalf of a mobile client.
      - Send this certificate back to the device.

   For details, see the Afaria server configuration documentation.
4. The IT administrator generates and distributes to end users the Common Name and Challenge Code required by the Certificate Authority configured for the Afaria portal package.

For complete details on Afaria provisioning, see *Provisioning with Afaria* in *System Administration*.

See also

- *Security Artifacts That Require Provisioning* on page 132
- *Provisioning Methods by Application Type* on page 133
- *Provisioning with Unwired Server* on page 136

Configuring Application Code Properties for HTTPS Connections

Application client properties can be set by the mobile application developer to connect to the secure Unwired Server port.

Ensure the application code uses the HTTPS protocol, port, and stream parameters.

- Unwired Server port – 2481.
- Protocol – HTTPS (equivalent to the MobiLink Stream Type).
- Stream Parameter – `trusted_certificates=mypublic_cert.crt`

Provisioning with Unwired Server

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

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

1. Install the device client application to the device.

2. Connect to the corporate LAN upon which Unwired Platform is installed.
3. Use a device connection that connects directly to Unwired Server. Alternatively, you can also connect using the Relay Server settings, but only if it is accessible from the corporate LAN.
4. The messaging service on Unwired Server seeds the client with the public key.

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

See also

- *Security Artifacts That Require Provisioning* on page 132
- *Provisioning Methods by Application Type* on page 133
- *Provisioning With Afaria: Security Considerations* on page 135

Establishing Encrypted Application Connections

Synchronization and messaging connection are encrypted by default. However, for replication connections that use E2EE, the client must be configured correctly to establish connections to the correct HTTP or HTTPS port.

See also

- *Provisioning Security Artifacts* on page 132

Connecting to the TLS Relay Server Port with Client APIS

(Applies only to Windows Mobile and Android devices with replication packages). With the Relay Server environment configured, developers can set appliction client properties to connect to it via the correct port using the TLS protocol.

Note: If Relay Server uses HTTPS and certificates, clients other than replication may not be able to connect: messaging applications only support HTTP, and hybrid workflow container applications for iOS support HTTPS — but not certificates.

1. Ensure the application code has been modified to use the correct TLS protocol, port, and stream parameters, for example:
   - Port – 443
   - Protocol – TLS
   - Stream parameter – `"url_suffix=/ias_relay_server/client/rs_client.dll/[SUP_FARM_ID];tls_type=RSA;trusted_certificates=rsa_root.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.

These certificates are personal certificate for the specific user. Typically this file type is not included as part of the application, but separately-installed by the user. In this case, ensure the application prompts the user for the filename and password of that certificate and save it to this parameter.

2. Make the `rsa_root.crt`, and`id_client.pem` (if it is not a personal file the user defines) available for the application on the device. They can be included in the application or deployed separately.

Connecting to the SSL Relay Server Port

Android and Windows Mobile replication clients should connect to Unwired Server through a Relay Server on the DMZ.

Prerequisites

When using E2EE over SSL, certificates need to be created and distributed to both Unwired Server and clients.

Task

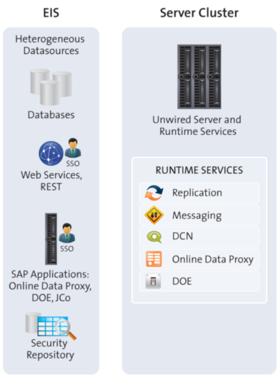
Only Windows Mobile can enable E2EE over SSL because these clients use an UltraLite client database. To enable E2EE:

1. Configure application connection code properties to connect to Relay Server with the correct combination of properties.
2. Generate and deploy files that reference this configuration.

For details see *Enable End-to-End Encryption (E2EE) Using SSL* in *Developer Guide: Windows Mobile Native Applications*.

CHAPTER 8 EIS Security

Secure interactions with EIS data sources. DCNs can be securely communicated to Unwired Server. Otherwise, appropriately secure the connections Unwired Server makes to each heterogeneous data source.



Securing Data Change Notifications

If you use data change notifications (DCN) to notify Unwired Server of EIS changes, secure the communication stream to avoid packet sniffing or data tampering. Use Sybase Control Center to configure the DCN stream after you have created SSL certificates.

1. *Preparing SSL Certificates for DCNs*

   DCN, which uses HTTP Basic authentication, uses a Base64-encoded username:password field that can be intercepted by network sniffers. Encrypt DCN communications and always use HTTPS to send DCNs. HTTPS requires the DCN sender import the Unwired Server certificate (or that of its CA signer) into its local equivalent of a truststore.
2. *Creating and Enabling a DCN Security Profile*

   An administrator must enable and configure the HTTPS port for DCN connections as part of a security profile so that developers can construct callouts from the EIS backend to send Unwired Server data change notification (DCN) requests.
3. *Enabling Authorization for Data Change Notification CDB Insertions*

   (Not applicable to Online Data Proxy) All DCN requests are authorized by checking if the user making the request is in the “SUP DCN User” role within the security configuration of the request’s target package.

See also

- *Unwired Server and EIS Communications* on page 7

Preparing SSL Certificates for DCNs

DCN, which uses HTTP Basic authentication, uses a Base64-encoded username:password field that can be intercepted by network sniffers. Encrypt DCN communications and always use HTTPS to send DCNs. HTTPS requires the DCN sender import the Unwired Server certificate (or that of its CA signer) into its local equivalent of a truststore.

In Sybase Control Center, configure DCN to use HTTPS by creating a security profile and enabling the HTTPS listener for DCN to use the security profile configured.

See also

- *Creating and Enabling a DCN Security Profile* on page 141

Creating and Importing Self-Signed Certificates in Development Environments

Use a self-signed certificate generated from the Java SDK **keytool** utility, andimports it into the Unwired Platform for DCN encryption testing.

Prerequisites

To use the **keytool** utility, set the `JAVA_HOME` environment variable to the JDK directory used by Unwired Platform, in addition to defining `%JAVA_HOME%\bin` as a the path variable. to the path variable, because keytool utility needs this setting.

Task

1. Change directory to *<UnwiredPlatform_InstallDir>*`\Repository\Security`, and run this command:

   ```
   keytool -genkey -alias dcn -keypass changeit -keyalg RSA -keysize
   1024 -validity 3650 -keystore dcn.jks
   -storepass chageit
   ```

   Follow the prompts to generate a public-private keypair for the Acme organization.

Prompt	Value Entered
Name	`Admin's name`
Organization	`Your company name`
Organizational Unit	`Development`
City	`Your city`
State/Province	`Your location`
Country Code	`Your country`

2. Use **keytool** to import the keystore to the destination keystore by using this command:

   ```
   keytool -importkeystore -destkeystore keystore.jks -deststorepass
   changeit -srckeystore dcn.jks
   -srckeypass changeit -alias DCN
   ```

3. Use **keytool** to export just the public key to the local disk.

   ```
   keytool -keystore keystore.jks -storepass changeit -alias DCN -
   export -file C:\temp\dcn.crt
   ```

Creating and Enabling a DCN Security Profile

An administrator must enable and configure the HTTPS port for DCN connections as part of a security profile so that developers can construct callouts from the EIS backend to send Unwired Server data change notification (DCN) requests.

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. Select **SSL Configuration**.
5. Create a new security profile by entering these values in an empty row of the table.

   For self-signed certificates, the alias value is the same value set with the **keytool** -alias property.

Column Name	Value
Security Profile	*`MyDCNprofileName`*
Certificate Alias	*`aliasUsed`*
Authentication	• Choose `intl` for crypto algorithms with weaker key strengths that can be used internationally. • Choose `domestic` for stronger crypto algorithms that can be used in the United States. • Choose `strong` for the strongest crypto algorithms available. If you choose this option you must download and install a different JCE from Oracle. Otherwise, this profile cannot be used.

6. On the General tab, click the **Communication Ports**.
7. Enable the listener for DCN:

   - If you are using Unwired Platform with MBOs, in the **SSL Profile** column for the port number 8001, select **Enabled**, then choose **dcn_profile**.

- If you are using Unwire Platform with OData and Gateway, enter a new port of 8004, select **Enabled** then choose *MyDCNprofileName*

8. Restart the Unwired Server.
9. Deploy a package that contains an application that uses DCN and test the DCN settings with it.

Next

Share connection properties with the development team.

See also

- *Preparing SSL Certificates for DCNs* on page 140

Enabling Authorization for Data Change Notification CDB Insertions

(Not applicable to Online Data Proxy) All DCN requests are authorized by checking if the user making the request is in the “SUP DCN User” role within the security configuration of the request’s target package.

Often the DCN user is not a “real” user in the customer’s security systems, but rather an artificial technical user whose credentials are simply a shared secret between the EIS developer writing the DCN client code, and the platform administrator.

If your DCN user is not a "real" user, you may also consider using the PreconfiguredUserLoginModule as an alternative to HTTPS Basic.

1. In Sybase Control Center, ensure that the MBO package that is the target of DCN uses a security configuration that includes either HTTP Basic authentication or PreconfiguredUser.

   Note: Sybase strongly recommends you do not use the "admin" security configuration for MBO packages. Keep your admin users and mobile users separate.

2. If you are using HTTPS Basic, thensure that the user name specified in the HTTPS Basic authentication response needs is formatted as:

   ```
   dcnSubject@security_configuration
   ```

   This format ensures that authentication/authorization happens in the correct security context.

   Note: In some ActiveDirectory configurations, the username may have the form user@activeDirectoryDomain. If a DCN user is fred@acme.com, and the security configuration is named "myCustomConfig", then the username specified for DCN request must be "fred@acme.com@myCustomConfig".

3. If you are using PreconfiguredUser:

   a) Stack this module in an appropriate order with your other LoginModules.

b) Set the username and password to something that you can share with the EIS developers who are writing the DCN sending logic, and set the Roles property to "SUP DCN User".
c) Ensure you set Control Flag to sufficient.

See also

- *Assigning Providers to a Security Configuration* on page 61
- *Stacking Providers and Combining Authentication Results* on page 98
- *HTTP Authentication Security Provider* on page 60
- *HTTP Basic Authentication Properties* on page 172

CHAPTER 9 Security Monitoring and Issue Detection

Identify, analyze, and resolve current and potential Unwired Platform security problems.

Tools and Diagnostic Methodologies

Sybase recommend administrators use of multiple diagnostic sources. Monitoring data can effectively supplement obscure and abbreviated debug/syslog output.

Platform Security Monitoring

Active security monitoring is about catching small problems before they turn into serious issues. It's also about taking proactive steps to protect your Unwired Platform against unnecessary risks. Use Sybase Control Center to help you proactively maintain the defense mechanisms you diligently implemented for your mobile components, data, and resources.

Current, historical, and performance-based tabs allow you to diagnose security health and issues for user support, troubleshooting, and security performance tracking.

Reviewing System Monitoring Data

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

1. Open and log in to Sybase Control Center.
2. In the left navigation pane, select **Monitoring**.
3. In the right administration pane, select one of the following tabs according to the type of monitoring data you want to view:

    - **Security Log**
    - **Replication**
    - **Messaging**
    - **Queue**
    - **Data Change Notifications**
    - **Device Notifications**
    - **Package Statistics**
    - **User Statistics**
    - **Cache Statistics**

Security Log Statistics

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

User security data falls into these categories:

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

Common Analysis Scenarios

Use these scenarios to walk you through typicaly security issues that may require a particular assessment path.

Access Denied Analysis

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

Checking the Security Log

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

1. In Sybase Control Center, click the Monitoring node.
2. Click **Security Log**.
3. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the specified time frame.
4. Click the **Result** column to sort rows by result type.
5. Locate any authentication failures or access denied events that are logged for the user who reported the error.

6. If you find any errors in the result column, check package names and security configurations. Then check to see if a similar result is reported for other user names. Also verify if the error persists; a heavily loaded service could cause a transient error. Transient errors are generally resolved retrying the connection.

Next

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

Validating Security Setup

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

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

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

APPENDIX A Security Reference

Configuration Files

Configuration files hold the initial settings for various Unwired Platform components or subcomponents.

All Unwired Platform components read their configuration files at start-up. Some components check configuration files for changes periodically. Administrators can instruct Unwired Server to reread configuration files, and apply changes to the current process. However, some configuration changes require you to restart the Unwired Server.

Role Mapping (roles-map.xml) Configuration File

Use the `<UnwiredPlatform_InstallDir>\SCC-XX\conf\roles-map.xml` to map roles for Sybase Control Center.

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. Typically, Sybase recommends that you only use the Delegation Login Module and only map roles for this login :

```
<?xml version="1.0" encoding="UTF-8"?>
<roles-map>
 <uaf-roles>
   <role name="uaAgentAdmin" description="Agent administrator
role" />
   <role name="uaPluginAdmin" description="Plugin administrator
role" />
   <role name="uaOSAdmin" description="Operation system
administrator role" />
   <role name="uaASEAdmin" description="ASE administrator role" />
   <role name="uaUser" description="User role" />
   <role name="uaGuest" description="Guest role" />
   <role name="uaAnonymous" description="Anonymous role" />
   <role name="sccAdminRole" description="SCC Administrator Role" />
   <role name="sccOperRole" description="SCC Operator Role" />
   <role name="sccUserRole" description="SCC User Role" />
   <role name="sccGuestRole" description="SCC Guest Role" />
   <role name="jmxDirectAccess" description="JMX Direct Access
Role" />
 </uaf-roles>
 <security-modules>

   <module name="SUP Delegation Login Module">
     <role-mapping modRole="SUP Administrator"
uafRole="uaAnonymous,uaAgentAdmin,uaPluginAdmin,sccAdminRole,sccUse
rRole,sccOperRole,sccGuestRole,jmxDirectAccess"/>
     <role-mapping modRole="SUP Domain Administrator"
uafRole="uaAnonymous,uaAgentAdmin,uaPluginAdmin,sccUserRole,sccOper
Role,sccGuestRole,jmxDirectAccess"/>
   </module>
   <module name="Anonymous Login Module">
     <role-mapping modRole="uaAnonymous" uafRole="uaAnonymous" />
   </module>
 </security-modules>
</roles-map>
```

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.

CSI Configuration (csi.properties) for Sybase Control Center

Use `<`*`SCC_HOME`*`>\conf\csi.properties` to configure security for Sybase Control Center logins.

A complete and correct `csi.properties` file for Unwired Platform includes these entries (login modules, authorizers, and attributers), with the Delegation Module being the only supported login module for Sybase Control Center:

```
## This file defines the default CSI configuration

###########################################
## login modules ! ##
###########################################
## Anonymous Login Module
CSI.loginModule.
1.provider=com.sybase.ua.services.security.anonymous.AnonymousLogin
Module
CSI.loginModule.1.controlFlag=sufficient
CSI.loginModule.1.options.moduleName=Anonymous Login Module
CSI.loginModule.1.options.roles=uaAnonymous

###########################################
## authorizers ##
###########################################
## XML Authorizer
CSI.authorizer.
1.provider=com.sybase.ua.services.security.xml.XMLAuthorizer
```

```
###########################################
## attributers ##
###########################################
## XML Attributer
CSI.attributer.
1.provider=com.sybase.ua.services.security.xml.XMLAttributer

## SUP PreConfiguredUser Delegation Login Module
CSI.loginModule.2.options.moduleName=SUP Delegation Login Module
CSI.loginModule.
2.provider=com.sybase.ua.services.security.sup.SUPDelegateLoginModu
le
CSI.loginModule.2.controlFlag=sufficient
```

Security Provider Configuration Properties

Security providers implement different properties, depending on whether or not they support authentication, authorization, audit, or attributionauthentication or authorization.

Platform administrators can configure application security properties in the Sybase Control Center for Unwired Platform console. These properties are then transcribed to an XML file in the `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\Repository\CSI\` directory. A new section is created for each provider you add.

LDAP Configuration Properties

(Not applicable to Online Data Proxy) 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.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.

- **LDAPLoginModule**– provides authentication services. Through appropriate configuration, you can enable certificate authentication in **LDAPLoginModule**.

- (Optional)**LDAPAuthorizer** or **RoleCheckAuthorizer** – provide authorization services for **LDAPLoginModule**. **LDAPLoginModule** works with either authorizer. In most production deployments, you must always 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**.

  The RoleCheckAuthorizer is used with every security configuration but does not appear in Sybase Control Center.

  Use **LDAPAuthorizer** only when **LDAPLoginModule** is not used to perform authentication, but roles are still required to perform authorization checks against the LDAP data store. If you use **LDAPAuthorizer**, always explicitly configure properties; for it cannot share the configuration options specified for the **LDAPLoginModule**.

Use this table to help you configure properties for one or more of the supported LDAP providers. 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). 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 of "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.

Property	Default Value	Description
AuthenticationFilter	For most LDAP servers: `(&amp;(uid={uid})(objectclass=person))` or For Active Directory email lookups: `(&amp;(userPrincipalName={uid})(objectclass=user))[ActiveDirectory]` For Active Directory Windows username lookups: `(&amp;(sAMAccountName={uid})(objectclass=user))` **Note:** Please note these restrictions when using this property to authenticate Sybase Control Center administration use cases only: • Do not use special characters (for example, `, = : ' " * ? &`) in user names identified with this property. • Do not use Chinese or Japanese characters in the user name	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
	or passwords of this property.	
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 property is not configured, the value for Default-SearchBase 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.
		
		Sybase recommends encrypting passwords and provides a password encryption utility for the purpose. If you encrypt BindPassword, include `encrypted=true` in the line that sets the option. For example:
		`<options name="BindPassword" en-crypted="true" value="1snjikf-wregfqr43hu5io..."/>`
		
		If you do not encrypt BindPassword, the option might look like this:
		`<options name="BindPassword" value="s3cr3T"/>`

RoleSearchBase	none	The search base used to retrieve lists of roles. If this property is not configured, the value for De-faultSearchBase is used.

Property	Default Value	Description
RoleFilter	For SunONE/iPlanet: `(&amp;(objectclass=ldapsubentry) (objectclass=nsroledefinition))` For Netscape Directory Server: `(|(objectclass=groupofnames) (objectclass=groupofuniquenames))` 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 and this property is not initialized, no roles are available.
RoleMemberAttributes	For Netscape Directory Server and OpenLDAP Server: member,uniquemember	A comma-separated list of role attributes from which LDAP derives the DNs of users who have this 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 of the role entry used as the role name in Unwired Platform. This is the role name displayed in the role list or granted to the authenticated user.
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.

Property	Default Value	Description
SkipRoleLookup	false	Set this property to true to grant the roles looked up using the attributes specified by the property UserRoleMembershipAttributes without cross-referencing them with the roles looked up using the RoleSearchBase and RoleFilter.
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. **Note:** If SkipRoleSearch property is set to true, then these comma-delimited values will not be cross-referenced with the roles retrieved in the role search base and role search filter. See *Skipping LDAP Role Lookups (SkipRoleLookup)*. **Note:** If you use nested groups with ActiveDirectory, you must set this property to "tokenGroups". See *Using LDAP Nested Groups and 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".
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.

Property	Default Value	Description
UseUserAccountContro-lAttribute	For ActiveDirectory: true	When this property is set to true, the UserAc-countControl attribute is used for detecting disa-bled user accounts, account expirations, password expirations and so on. ActiveDirectory also uses this attribute to store the above information.
controlFlag	optional	When you configure multiple Authentication pro-viders, use controlFlag for each provider to con-trol how the authentication providers are used in the login sequence. **Note:** controlFlag is a generic login module op-tion rather than an LDAP configuration property.
EnableLDAPConnection-Trace	None	Enables LDAP connection tracing. The output is logged to a file in temp directory. The location of the file is logged to the server log.

See also

- *LDAP Security Provider* on page 37
- *Adding a Production-Grade Provider* on page 35
- *Stacking Providers and Combining Authentication Results* on page 98

NTProxy Configuration Properties

(Not applicable to Online Data Proxy) 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 5. Authentication properties

Properties	Default Value	Description
Extract Domain From User-name	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.

Properties	Default Value	Description
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.

NoSec Configuration Properties

A NoSec provider offers pass-through security for Unwired Server. In development environments, you can apply a NoSec security provider for authentication, authorization, and attribution authentication and authorization modules. However, never use this provider in production environments — either for administration, or device user authentication.

- The NoSecLoginModule class provides open authentication services
- The NoSecAuthorizer class provides authorization services
- The NoSecAttributer provides attribution services

However, you need to configure only authentication properties for a NoSec provider.

Table 6. Authentication properties

Property	Default Value	Description
useUsernameAsIdentity	true	If this option is set to true, the user name supplied in the call-back is set as the name of the principal added to the subject.
identity	nosec_identity	The value of this configuration option is used as the identity of the user if either of these conditions is met: • No credentials were supplied. • The useUsernameAsIdentity option is set to false.
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.

Certificate Authentication Properties

Add and configure authentication provider properties for CertificateAuthenticationLoginModule, or accept the default settings.

Note: This provider cannot be used for administrative security (in the "admin" security configuration).

Table 7. CertificateAuthenticationLoginModule properties

Property	Description
Implementation class	The fully qualified class that implements the login module. `com.sybase.security.core.CertificateAuthenticationLoginModule` is the default class.
Provider type	`LoginModule` is the only supported value.
Control flag	Determines how success or failure of this module affects the overall authentication decision. `optional` is the default value.
Clear password	(Optional) If true, the login module clears the user name and password from the shared context. The default is false.
Store password	(Optional) If true, the login module stores the user name and password in the shared context. The default is false.
Try first password	(Optional) If true, the login module attempts to retrieve user name and password information from the shared context, before using the callback handler. The default is false.
Use first password	(Optional) If true, the login module attempts to retrieve the user name and password only from the shared context. The default is false.
Enable revocation checking	(Optional) Enables online certificate status protocol (OCSP) certificate checking for user authentication. If you enable this option, you muse enable OCSP in Unwired Server. This provider uses the values defined as part of the SSL security profile. Revoked certificates result in authentication failure when both of these conditions are met: • revocation checking is enabled • OCSP properties are configured correctly
Regex for username certificate match	(Optional) By default, this value matches that of the certificates common name (CN) property used to identify the user. If a mobile application user supplies a user name that does not match this value, authentication fails.

Property	Description
Trusted certificate store	(Optional) The file containing the trusted CA certificates (import the issuer certificate into this certificate store). Use this property and `Store Password` property to keep the module out of the system trust store. The default Unwired Server system trust store is `<UnwiredPlatform_InstallDir\Servers\UnwiredServer\Repository\Securitytruststore\truststore.jks`. If you do not specify a store location:: • Unwired Server checks to see if a store used by the JVM (that is, the one defined by the `javax.net.ssl.trustStoreType` system property. • If the system property is not defined, then this value is used: `${java.home}/lib/security/jssecacerts` • If that location also doesn't exist, then this value is used: `${java.home}/lib/security/cacerts` **Note:** This property is required only if Validate certificate path is set to true.
Trusted certificate store password	(Optional) The password required to access the trusted certificate store. For example, import the issuer of the certificate you are trying to authenticate into the shared JDK cacerts file and specify the password using this property. **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStorePassword` property.
Trusted certificate store provider	(Optional) The keystore provider. For example, "SunJCE." **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStoreProvider` property. If it is not defined, then the most preferred provider from the list of registered providers that supports the specified certificate store type is used.

Property	Description
Trusted certificate store type	(Optional) The type of certificate store. For example, "JKS." **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStore` property. If this value is not defined, then default value is the keystore type as specified in the Java security properties file, or the string "jks" (Java keystore) if no such property exists.
Validate certificate path	If true (the default), performs certificate chain validation of the certificate being authenticated, starting with the certificate being validated. Verifies that the issuer of that certificate is valid and is issued by a trusted certificate authority (CA), if not, it looks up the issuer of that certificate in turn and verifies it is valid and is issued by a trusted CA. In other words, it builds up the path to a CA that is in the trusted certificate store. If the trusted store does not contain any of the issuers in the certificate chain, then path validation fails. For information about adding a certificate to the truststore, see *Using Keytool to Generate Self-Signed Certificates and Keys* in *Security*.

See also

- *Certificate Security Provider* on page 60
- *SAP SSO Token Security Provider* on page 59
- *HTTP Authentication Security Provider* on page 60
- *Using Keytool to Generate Self-Signed Certificates and Keys* on page 44

Certificate Validation Properties

Add and configure provider properties for CertificateValidationLoginModule, or accept the default settings. CertificateValidationLoginModule can be used in conjunction with other login modules that support certificate authentication (for example, LDAPLoginModule) by configuring CertificateValidationLoginModule before the login modules that support certificate authentication.

You can only use this provider to validate client certificates when an HTTPS listeners is configured to use mutual authentication.

Table 8. CertificateValidationLoginModule properties

Property	Description
Implementation class	The fully qualified class that implements the login module. `com.sybase.security.core.CertificateValidationLoginModule` is the default class.

crl.[index].uri	Specifies the universal resource identifier for the certificate revocation list (CRL). Multiple CRLs can be configured using different values for the index. The CRLs are processed in index order. For example:
	`crl.1.uri=http://crl.verisign.com/ThawtePersonalFreemailIssuingCA.crl`
	`crl.2.uri=http://crl-server/`

Provider type	`LoginModule` is the only supported value.
Validated certificate is identity	(Optional) Determines if the certificate should be set the authenticated subject as the user ID. If the CertificateValidationLoginModule is used in conjunction with other login modules that establish user identity based on the validated certificate, set this value to `false`. If you are implementing this provider with a DCN security configuration, and it's also not used with SSO, then set this property to `true`.`False` is the default value.
Enable revocation checking	(Optional) Enables online certificate status protocol (OCSP) certificate checking for user authentication. If you enable this option, you muse enable OCSP in Unwired Server. This provider uses the values defined as part of the SSL security profile. Revoked certificates result in authentication failure when both of these conditions are met: • revocation checking is enabled • OCSP properties are configured correctly

Property	Description
Trusted certificate store	(Optional) The file containing the trusted CA certificates (import the issuer certificate into this certificate store). Use this property and `Store Password` property to keep the module out of the system trust store. The default Unwired Server system trust store is `<UnwiredPlatform_InstallDir\Servers\UnwiredServer\Repository\Securitytruststore\truststore.jks`. If you do not specify a store location:: • Unwired Server checks to see if a store used by the JVM (that is, the one defined by the `javax.net.ssl.trustStoreType` system property. • If the system property is not defined, then this value is used: `${java.home}/lib/security/jssecacerts` • If that location also doesn't exist, then this value is used: `${java.home}/lib/security/cacerts` **Note:** This property is required only if Validate certificate path is set to true.
Trusted certificate store password	(Optional) The password required to access the trusted certificate store. For example, import the issuer of the certificate you are trying to authenticate into the shared JDK cacerts file and specify the password using this property. **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStorePassword` property.
Trusted certificate store provider	(Optional) The keystore provider. For example, "SunJCE." **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStoreProvider` property. If it is not defined, then the most preferred provider from the list of registered providers that supports the specified certificate store type is used.

Property	Description
Trusted certificate store type	(Optional) The type of certificate store. For example, "JKS." **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStore` property. If this value is not defined, then default value is the keystore type as specified in the Java security properties file, or the string "jks" (Java keystore) if no such property exists.
Validate certificate path	If true (the default), performs certificate chain validation of the certificate being authenticated, starting with the certificate being validated. Verifies that the issuer of that certificate is valid and is issued by a trusted certificate authority (CA), if not, it looks up the issuer of that certificate in turn and verifies it is valid and is issued by a trusted CA. In other words, it builds up the path to a CA that is in the trusted certificate store. If the trusted store does not contain any of the issuers in the certificate chain, then path validation fails. For information about adding a certificate to the truststore, see *Using Keytool to Generate Self-Signed Certificates and Keys* in *Security*.

SAP SSO Token Authentication Properties

The SAPSSOTokenLoginModule has been deprecated, Use the HttpAuthenticationLoginModule when SAP SSO2 token authentication is required. This authentication module will be removed in a future release.

Table 9. SAPSSOTokenLoginModule properties

Property	Description
Implementation class	(Required) – the fully qualified class that implements the login module. `com.sybase.security.sap.SAPSSOTokenLoginModule` is the default class.
Provider type	(Required and read-only) – `LoginModule` is the only supported value.
Control flag	(Required) – `optional` is the default value. Determines how success or failure of this module affects the overall authentication decision.

Property	Description
SAP server URL	(Required) – the SAP server URL that provides the SSO2 token. This may or may not be the same server that authenticates the user. If providing and authenticating servers are different, you must import the SAP Token provider server certificate or one of its CA signers into the Unwired Server truststore in addition to that of the authenticating server to enable HTTPS communication. In environments where the servers are different, the basic flow is: **1.** Unwired Server passes credentials over HTTPS to the token granting service. **2.** An SSO2Token cookie is returned to Unwired Server. **3.** The SSO2Token flows to the authenticating server, which could be an SAP EIS or a server that hosts a Web service bound to SAP function modules. **Note:** The SAP Server URL must be configured to require BASIC authentication, not just FORM based authentication.
Clear password	(Optional) – if set to `True`, the login module clears the username and password in the shared context.
Disable server certificate validation	(Optional) – the default is False. If set to `True`, disables certificate validation when establishing an HTTPS connection to the SAP server using the configured URL. Set to `True` only for configuration debugging.
SAP server certificate	(Optional) – name of the file containing the SAP certificate's public key in `.pse` format. This is required only when token caching is enabled by setting a SAP SSO token persistence data store value.
SAP server certificate password	(Optional) – password used to access the SAP server certificate.

Property	Description
SAP SSO token persistence data store	(Optional) – JNDI name used to look-up the data source to persist the retrieved SSO2 tokens. Set to "jdbc/default" to store tokens in the Unwired Server CDB. If unconfigured, some caching is still done based on the "Authentication cache timeout interval" property associated with the security configuration setting. If you use the default setting, you do not need to set SAP SSO token persistence data store, SAP server certificate, SAP server certificate password, or Token expiration interval properties. To enable token caching through the SAPSSOTokenLogin-Module: **1.** Set the SAP SSO token persistence data store value to "jdbc/default." **2.** Download and install the SAP SSO2 token files. See *Installing the SAP SSO2Token Files on Unwired Server Hosts* in the *Security* guide. **3.** Specify the correct value for the SAP server certificate, SAP server certificate, SAP server certificate password and Token expiration interval properties.
Store password	(Optional) – if set to true, the login module stores the user-name/password in the shared context after successfully authenticating the user.
Token expiration interval	(Optional) – this property is ignored when the SAP SSO token persistence data store property is not configured. It specifies the token validity period, after which time a new token is retrieved from the SAP EIS. The default value is 120 seconds. Keep in mind that: • The "Token expiration interval" cannot exceed the "Token validity period", which is the amount of time defined in the back-end SAP server for which the token is valid. • The "Authentication cache timeout" property must be less than the "Token expiration interval" property value.

Property	Description
Try first password	(Optional) – if set to `True`, the login module attempts to retrieve the username/password from the shared context, before calling the callback handler.
Use first password	(Optional) – if set to `True`, the login module attempts to retrieve the username/password only from the shared context, and never calls the callback handler.
HTTP connection timeout interval	The value, in seconds, after which an HTTP(s) connection request to the EIS times out. If the HTTP connection made in this module (for either user authentication or configuration validation) does not have a time out set, and attempts to connect to an EIS that is unresponsive, the connection hangs, which could potentially cause Unwired Server to hang. Setting the timeout interval ensures authentication failure is reported without waiting for ever for the server to respond.

See also

- *Certificate Security Provider* on page 60
- *SAP SSO Token Security Provider* on page 59
- *HTTP Authentication Security Provider* on page 60

Preconfigured User Authentication Properties

The PreConfiguredUserLoginModule authenticates the Unwired Platform Administrator user whose credentials are specified during installations.

This login module is recommended only to give the Platform administrator access to Sybase Control Center so it can be configured for production use. Administrators are expected to replace this login module immediately upon logging in for the first time.

The PreConfiguredUserLoginModule:

- Provides role based authorization by configuring the provider com.sybase.security.core.RoleCheckAuthorizer in conjunction with this authentication provider.
- Authenticates the user by comparing the specified username/password against the configured user. Upon successful authentication, the configured roles are added as Principals to the Subject.

Table 10. PreConfiguredUserLoginModule properties

Property	Description
User name	A valid user name. Do not use any of these restricted special characters: `, = : ' " * ? &`.
Password	The encoded password hash value.
Roles	Comma separated list of roles granted to the authenticated user for role-based authorization. Platform roles include "SUP Administrator" and "SUP Domain Administrator". Roles are mandatory for "admin" security configuration. For example, if you define "SUP Administrator" to this property, the login id in the created login module has Platform administrator privileges. **Note:** If you use other values, ensure you map Unwired Platform roles to the one you define here.

See also

- *Preconfigured User Login Security Provider* on page 36
- *Adding a Production-Grade Provider* on page 35
- *Mapping Unwired Platform Logical Roles to Physical Roles* on page 38

HTTP Basic Authentication Properties

The HttpAuthenticationLoginModule provider authenticates the user with given credentials (user name and password) against the secured Web server (SWS) using a GET against a URL that requires basic authentication, and can be configured to retrieve a cookie with the configured name and add it to the JAAS subject to facilitate single sign-on (SSO) or network edge authentication.

This provider can be configured for authenticating the user when:

- using only the specified username/password
- using only the specified client value(s)
- first attempting token authentication and if it fails, reverting to basic authentication using the supplied username/password. This could be helpful when using the same security configuration for authenticating users with a token, such as device users hitting network edge, and when DCN requests from within a firewall present only the username/password but no token.

Note: The HttpAuthenticationLoginModule allows token validation by connecting to an HTTP server capable of validating the token specified in the HTTP header and cookie set in the session.

Table 11. HttpAuthenticationLoginModule configuration options

Configuration Option	Default Value	Description
URL	None	The HTTP(S) URL that authenticates the user. For single sign-on, this is the server URL from which Unwired Server acquires the SSO cookie/token.
Disable certificate validation	False	(Optional) The default is false. If set to true , disables certificate validation when establishing an HTTPS connection to the SWS using the configured URL. Set to true only for configuration debugging.
SSO cookie name	None	(Optional) A name of the cookie that is set in the session between the LoginModule and the SWS and holds the SSO token for single sign-on. The provider looks for this cookie in the connection to the SWS. If found, it is added to the authenticated subject as a named credential. The authentication provider ignores the status code when a SSO cookie is found in the session. If the cookie is found, authentication succeeds regardless of the return status code.
Roles HTTP header	None	(Optional) The name of an HTTP header that the server may return. The header value contains a comma-separated list of roles to be granted.
Successful connection status code	200	HTTP status code interpreted as success when connection is established to the SWS.

Configuration Option	Default Value	Description
HTTP connection timeout interval	1 minute	The value, in seconds, after which an HTTP(s) connection request to the Web-based authentication service times out. If the HTTP connection made in this module (for either user authentication or configuration validation) does not have a time out set, and attempts to connect to a Web-based authentication service that is unresponsive, the connection hangs, which could potentially cause Unwired Server to hang. Setting the timeout interval ensures authentication failure is reported without waiting for ever for the server to respond.
SendClientHttpValuesAs	None	Comma separated list of strings that indicate how the ClientHttpValuesToSend should be sent to the HTTP server. For example: `SendClientHttpValuesAs=header:`*`header_name`*`, cookie: `*`cookie_name`* **Note:** If the user should be authenticated only using the supplied username/password, then this property does not apply.

Configuration Option	Default Value	Description
ClientHttpValuesToSend		A comma separated list of client HTTP values that should be sent to the HTTP server. For example: `ClientHttpValuesToSend=`*`client_personalization_key`*`,` *`client_cookie_name`* This property should be set if token authentication is used. Setting the property "ClientHttpValuesToSend" triggers token authentication. Unless TryBasicAuthIfTokenAuthFails is configured to true in conjunction with ClientHttpValuesToSend, only token authentication will be attempted. **Note:** If the user should be authenticated only using the supplied username/password, then this property does not apply.
SendPasswordAsCookie	None	Sends the password to the URL as a cookie with this name. If not specified, the password is not sent in a cookie. This property is normally used when there is a cookie-based SSO mechanism in use (for example, SiteMinder), and the client has put an SSO token into the password. The token can be propagated from the personalization keys and HTTP header/cookies to the SWS without impacting the password field.

Configuration Option	Default Value	Description
TryBasicAuthIfTokenAuth-Fails	False	Option that specifies if the provider should attempt basic authentication using the specified username/password credentials if token authentication is configured and it fails. This property is applicable only if token authentication is enabled. **Note:** If the user should be authenticated only using the supplied username/password, then this property does not apply.
UsernameHttpHeader	None	Http response header name that is sent back by the HTTP server with the username retrieved from the token. The retrieved username is added as a SecNamePrincipal upon successful authentication. **Note:** If the user should be authenticated only using the supplied username/password, then this property does not apply.

Configuration Option	Default Value	Description
regexForUsernameMatch	None	Regular expression to use for matching the supplied username with the username returned by the HTTP server in the UsernameHttpHeader. The string "{username}" in the regex is replaced with the specified username before using it. If specified, it is used to match the username retrieved from the UsernameHttpHeader to the username specified in the callback handler. It they do not match, it results in authentication failure. If they match, both the specified username and the retrieved username are added as SecNamePrincipals to the authenticated subject. **Note:** If the user should be authenticated only using the supplied username/password, then this property does not apply.

Configuration Option	Default Value	Description
TokenExpirationTimeHttpHeader	None	HTTP response header name that is sent back by the HTTP server with the validity period of the token in milliseconds from the start of January 1, 1970. If the header is returned in the HTTP response from the SWS, the token is cached for the duration it remains valid unless TokenExpirationInterval is also configured. If this response header is not returned with the token, it might result in unintended use of the token attached to the authenticated context even after it has expired. **Note:** If the user should be authenticated only using the supplied username/password, then this property does not apply.

Configuration Option	Default Value	Description
TokenExpirationInterval	0	Property to specify the interval in milliseconds to be deducted from the actual expiration time returned in TokenExpirationTimeHttpHeader. This ensures that the token credential retrieved from the authenticated session remains valid until it is passed to the SWS for single sign-on to access MBOs. **Note:** If the TokenExpirationTimeHttpHeader value returned by the SWS is less than the value configured for the TokenExpirationInterval property, it results in authentication failure. **Note:** If the user should be authenticated only using the supplied username/password, then this property does not apply.
CredentialName	None	Name to set in the authentication credential that contains the token returned in SSOCookieName. If this property is not configured, the SSOCookieName is set as the name of the token credential.

See also

- *Certificate Security Provider* on page 60
- *SAP SSO Token Security Provider* on page 59
- *HTTP Authentication Security Provider* on page 60
- *Enabling Authorization for Data Change Notification CDB Insertions* on page 142
- *Assigning Providers to a Security Configuration* on page 61
- *Stacking Providers and Combining Authentication Results* on page 98

Auditor Filter Properties Reference

(Not applicable to Online Data Proxy) Configure multiple resource classes when defining an auditor for a named security configuration.

Filter resource classes require a specific syntax. Based on that syntax, an audit token is supplied to the core CSI classes. This audit token identifies the source for core audit requests of operations, such as auditing the results for authorization decisions, authentication decisions, in addition to placing information such as active provider information into the audit trail. The audit records have their resource class prefixed by the prefix core. CSI core will able to audit a large number of items.

Syntax

Filter resource classes consist of one or more filter expressions that are delimited by parenthesis (()). Square brackets ([]) denote optional values. The syntax is:

```
[key1=value [,key2=value...]].
```

The allowed keys are: `ResourceClass`, `Action`, or `Decision`.

This table describes core auditable items:

Resource Class	Action	Description	Attributes
provider	activate	Called when a provider is activated by CSI. The Resource ID is the provider class name.	Generated unique provider identifier.
subject	authentication.provider	The result of a provider's specific authentication request. Depending on the other providers active, the actual CSI request for authentication may not reflect this same decision. **Note:** that this is not a provider-generated audit record. CSI core will generate this audit record automatically after receiving the provider's decision. The resource ID is not used.	• Provider identifier • Decision (yes, no) • Failure reason (if any) • Context ID

Resource Class	Action	Description	Attributes
subject	authentication	The aggregate decision after considering each of the appropriate provider's authentication decisions. This record shares the same request identifier as the corresponding authentication.provider records. The resource ID is Subject identifier if authentication successful.	• Decision (yes or no) • Context ID
subject	authorization.role.provider	The result of a provider's specific role authorization request. The resource ID is the subject ID.	• Provider identifier • Decision (yes, no or abstain) • Role name • Supplied subject identifier (if different from context subject) • Context ID
subject	authorization.role	The result of a resource-based authorization request. The resource ID is the subject ID.	• Resource name • The access requested Decision (yes, no or abstain) • Supplied subject identifier (if different from context subject) • Context ID
subject	authorization.resource	The aggregate decision authorization decision after considering each of the appropriate provider's authorization decision. The resource ID is the subject ID.	• Resource name • Access requested Decision (yes, no) • Supplied subject identifier (if different from context subject) • Context ID

Resource Class	Action	Description	Attributes
subject	logout	Generated when an authenticated context is destroyed. The resource ID is the subject ID.	Context ID
subject	create.provider	Provider-level record issued for anonymous self registration requests. The resource ID is the subject identifier.	• Provider identifier • Decision • Subject attributes
subject	create	Aggregate, generated when an anonymous self-registration request is made. The resource ID is the subject identifier.	• Decision • Subject attributes
subject	authorization.resource	The aggregate authorization decision, which is made after considering each of the appropriate provider's result.The resource ID is the subject ID.	• Resource ID • Access requested • Decision (yes, no) • Subject ID supplied (if different from context subject) • Context ID

Examples

- **Example 1 –** enable auditing of all of the CSI core resource classes that involve a deny decision:

  ```
  (ResourceClass=core.*,Decision=Deny)
  ```

- **Example 2 –** enable auditing for all core resource classes where the action is the subject modification action:

  ```
  Resource=core.*,Action=subject.modify.*)
  ```

Certificate and Key Management Utilities

Use the certificate management utilities to encrypt Unwired Server ports.

Launch these utilities from the command line; the certificate and key management utilities are not available from any other administration tool.

Use **createcert** and **createkey** for MobiLink and Ultralite server/client purposes (specific for replication payload protocol packages). For all other purposes, use **keytool** or the PKI system deployed to your environment.

Certificate Creation (createcert) Utility

Generates X.509 certificates or signs pregenerated certificate requests. This utility is located in `<UnwiredPlatform_InstallDir>\Servers\SQLAnywhereXX\BINXX`.

You may choose to purchase certificates from a third party. These certificate authorities (CAs) provide their own tools for creating certificates. You can use **createcert** to create certificates for development and testing; you can also use it for production certificates.

Syntax

```
createcert [options]
```

Parameters

- **[*options*] –** these options are available through the **createcert** utility:

Option	Description
`-r`	Creates a PKCS #10 certificate request. **createcert** does not prompt for a signer or any other information used to sign a certificate.
`-s <filename>`	Signs the PKCS #10 certificate request that is in the specified file. The request can be DER or PEM encoded. **createcert** does not prompt for key generation or subject information.

Note: To create a signed certificate, use **createcert** without options. To break the process into two separate steps, for example so one person creates a request and another person signs it, the first person can run **createcert** with `-r` to create a request and the second person can sign the request by running **createcert** with `-s`.

When you run **createcert**, you are asked for all or some of this information, depending on whether you specified `-r`, `-s`, or neither.

- `Choose encryption type` – select RSA.
- `Enter RSA key length (512-16384)` – this prompt appears only if you chose RSA encryption. Specify a length between 512 bits and 16384 bits.
- Subject information – enter this information, which identifies the entity:
  - `Country Code`
  - `State/Province`
  - `Locality`
  - `Organization`

    - `Organizational Unit`
    - `Common Name`
- (Optional) `Enter file path of signer's certificate` – supply a location and file name for the signer's certificate. If you supply this information, the generated certificate is a signed certificate. If you do not supply this information, the generated certificate is a self-signed root certificate.
- `Enter file path of signer's private key` – supply a location and file name to save the private key associated with the certificate request. This prompt appears only if you supplied a file in the previous prompt.
- `Enter password for signer's private key` – supply the password that was used to encrypt the signer's private key. Supply this password only if the private key was encrypted.
- (Optional) `Serial number` – supply a serial number. The serial number must be a hexadecimal string of 40 digits or less. This number must be unique among all certificates signed by the current signer. If you do not supply a serial number, **createcert** generates a GUID as the serial number.
- `Certificate will be valid for how many years (1-100)` – specify the number of years for which the certificate is valid. After this period, the certificate expires, along with all certificates it signs.
- `Certificate Authority (y)es or (n)o` – indicate whether this certificate can be used to sign other certificates. The default value is no.
- `Key usage` – supply a comma-separated list of numbers that indicate the ways in which the certificate's private key can be used. The default, which depends on whether the certificate is a certificate authority, should be acceptable for most situations.
- `File path to save request` – this prompt appears only if you specify the -r option. Supply a location and file name for the PCKS #10 certificate request. Supply a location and file name in which to save the certificate. The certificate is not saved unless you specify a location and file name.
- `Enter file path to save private key` – supply a location and file name in which to save the private key. Enter a password to protect private key. Optionally, supply a password with which to encrypt the private key. If you do not supply a password, the private key is not encrypted. This prompt appears only if you supplied a file in the previous prompt.
- `Enter file path to save identity` – supply a location and file name in which to save the identity. The identity file is a concatenation of the certificate, signer, and private key. This is the file that you supply to the server at start-up. If the private key was not saved, **createcert** prompts for a password to save the private key. Otherwise, it uses the password provided earlier. The identity is not saved unless you provide a file name. If you do not save the identity file, you can manually concatenate the certificate, signer, and private key files into an identity file.

Examples

- **Example 1: –** creates a self-signed certificate. No file name is provided for the signer's certificate, which makes it a self-signed root certificate.

```
<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers
\SQLAnywhereXX\BINXX>createcert
SQL Anywhere X.509 Certificate Generator Version xx.xx.xx.xx
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
State/Province: CA
Locality: Dublin
Organization: MyCompanyCA
Organizational Unit: PTO
Common Name: MyCompanyCA
Enter file path of signer's certificate:
Certificate will be a self-signed root
Serial number [generate GUID]:<enter>
Generated serial number: 3f52ee68c8604e48b8359e0c0128da5a
Certificate valid for how many years (1-100): 10
Certificate Authority (Y/N) [N]: Y
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [6,7]: <enter>
Enter file path to save certificate: rsa_root.crt
Enter file path to save private key: rsa_key.key
Enter password to protect private key: <MyPwd>
Enter file path to save identity: id.pem
```

- **Example 2: Generating an enterprise root certificate –** to generate an enterprise root certificate (a certificate that signs other certificates), create a self-signed root certificate with a CA. The procedure is similar to Example 1. However, the response to the CA prompt should be `yes` and choice for roles should be option `6,7` (the default).

```
Certificate Authority (Y/N) [N]: y
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [6,7]: 6,7
```

Key Creation (createkey) Utility

Creates an RSA key pairs for use with Unwired Server end-to-end encryption. This utility is located in `<`*`UnwiredPlatform_InstallDir`*`>\Servers\SQLAnywhere`*`XX`*`\BIN`*`XX`*.

Syntax

```
createkey
```

When you run **createkey**, you are prompted for this information:

- `Choose encryption type` – choose RSA.
- `Enter RSA key length (512-16384)` – this prompt appears only if you chose RSA encryption. Choose a length between 512 bits and 16384 bits.
- `Enter file path to save public key` – specify a file name and location for the generated PEM-encoded public key. This file is specified on the MobiLink client by the e2ee_public_key protocol option.
- `Enter file path to save private key` – specify a file name and location for the generated PEM-encoded private key. This file is specified on the MobiLink server via the e2ee_private_key protocol option.
- `Enter password to protect private key` – optionally, supply a password with which to encrypt the private key. The private key is not encrypted if you do not supply a password. This password is specified on the MobiLink server via the e2ee_private_key_password protocol option.

Examples

- **Example –** creates an RSA key pair:

  ```
  >createkey
  SQL Anywhere Key Pair Generator Version 11.0.0.2376
  Enter RSA key length (512-16384): 2048
  Generating key pair...
  Enter file path to save public key: rsa_key_public.key
  Enter file path to save private key: rsa_key_private.key
  Enter password to protect private key: pwd
  ```

Truststore and Keystore Properties

The `<`*`UnwiredPlatform_InstallDir`*`>\SCC-`*`XX`*`\services\Messaging\lib\eas\lib\Repository\Server\EmbeddedJMS\Instance\com\sybase\djc\server\ApplicationServer\EmbeddedJMS.properties` file contains properties for the truststore and keystore that you can configure. While Sybase Control Center uses the same keystore and truststore location as Unwired server, This file only configures the keystore/truststore for the Sybase Control Center Windows service.

Change the default properties for:

Property	Default	Description
keyStore	`<`*`UnwiredPlatform_InstallDir`*`/Servers/UnwiredServer/Repository/Security/keystore.jks`	The default location of the keystore used by Sybase Control Center.
keyStorePassword	changeIt	The password used to unlock the keystore.
trustStore	`<`*`UnwiredPlatform_InstallDir`*`/Servers/UnwiredServer/Repository/Security/truststore.jks`	The default location of the truststore used by Sybase Control Center. The truststore is used when Sybase Control Center makes an out-bound connection over SSL to another server with a server certificate. Sybase Control Center checks that the server certificate is in the truststore, or is signed by a CA certificate in the truststore.
trustStorePassword	changeIt	The password used to unlock the truststore.

Port Number Reference

Change Sybase Unwired Platform component port numbers after installation, if necessary.

Proceed with caution when changing port numbers because the change might impact other configuration files that point to that port. You need to be aware of the default Sybase Control Center port numbers so you do not accidentally use these ports when you change Sybase Unwired Platform ports. You can change some Sybase Control Center default ports, but, in some cases, you should not.

Note: To make Unwired Server port number changes, temporarily stop the other service consuming those ports. Use Sybase Control Center to make the changes, then restart Unwired Server.

Port	Description	Default Port	Instructions for Changing
Data tier (CDB) server	Port number for the data tier that manages transactions between the enterprise information system and mobile devices.	5200	Do not change the CDB port.
Management ports	IIOP port number on which the Unwired Server listens for Sybase Control Center administration requests.	2000 2001 for secure management (default)	Default is recommended. No change is required.
Data change notification (DCN)	Port number on which Unwired Server listens for DCN events.	8000 for HTTP 8001 for HTTPS	Configure in Sybase Control Center by expanding the Servers > *<ServerName>* folder and selecting Server Configuration. In the General tab, select the Communication Ports subtab and enter a new DCN port or secure DCN port, as required. See *Security Profiles* in in Sybase Control Center online help.

Port	Description	Default Port	Instructions for Changing
Synchronization	Port numbers on which Unwired Server synchronizes data between the enterprise information system and mobile devices. Messaging port uses a proprietary encryption method, so communication is always encrypted.	2480 for replication 5001 for messaging	Configure in Sybase Control Center by expanding the Servers > *<ServerName>* folder and selecting Server Configuration. In the Replication or Messaging tab, select the Synchronization Listener subtab and enter a new synchronization port, as required. **Note:** If there is a conflict for port 2480 or 2481, Unwired Server will not start, and yo cannot use Sybase Control Center to modify them. To correct the problem, you must temporarily stop the service that uses the conflicting port, then start Unwired Server. For replication payloads, see *Configuring Replication Properties* in Sybase Control Center online help. For messaging payloads, see *Configuring Messaging Properties* in in Sybase Control Center online help.
Advantage Database Server®	Port number for the messaging database.	6262	Sybase recommends that you do not change these ports.

Port	Description	Default Port	Instructions for Changing
Messaging server administration	Port number for the messaging service for Sybase messaging clients.	5100 for administration services	Cannot be changed in Sybase Control Center. Use the `<<UnwiredPlatform_InstallDir>>\Servers\Messaging Server\Bin\AdminWebServicesTool.exe` command line tool to change the messaging service Web service port. This tool has built in online help describing how to use the tool. From the command prompt run: `<UnwiredPlatform_InstallDir>\Servers\Messaging Server\Bin>AdminWebServicesTool.exe set=<port> restart`

Port	Description	Default Port	Instructions for Changing
Sybase Control Center	Additional default port numbers of which to be aware, when modifying port numbers.	9999 for default RMI agent port 2100 for default JMS messaging service port 3638 for default SCC repository database port 8282, 8283 for default Web container ports	• 9999 – default RMI agent port. The port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\RMI\service-config.xml` • 2100 – default JMS messaging service port. The port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\Messaging\service-config.xml` • 3638 – default SCC repository database port. The default port is set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\SccSADataserver\service-config.xml` • 8282, 8283 – default Web container ports. The default ports are set in: `<<UnwiredPlatform_InstallDir>>\SCC-XX\services\EmbeddedWebContainer\service-config.xml` Before you make any changes to these files, stop Sybase Control Center X.X service. Start the service after you complete the changes. If any of the subsystems fail to start, check the SCC `agent.log` for error messages.

Port	Description	Default Port	Instructions for Changing
Relay server	Port numbers on which Relay Server listens for requests.	80 for the HTTP port 443 for the HTTPS port	Change the value for the cluster in Sybase Control Center for Unwired Platform. You can then generate the file and transfer it to the corresponding Unwired Server host. See *Sybase Control Center > Configure > Configure Unwired Platform > Clusters >Relay Server*.
Unwired Platform reserved	Port numbers reserved for internal use by Unwired Platform components	4343 5500 27000 8002 2638	Do not use these special port numbers for any purpose. These ports differ from Windows reserved ports (1-1023). **Note:** Even if the installer does not detect a conflict at install time, Windows may later use ports in the 1024-64K range for other purposes. Read Microsoft documentation to determine how to reserve Unwired Platform ports. Otherwise, you may experience intermittent problems when starting up platform services due to Windows using these ports for some other purpose at the same time.

APPENDIX B

Glossary: Sybase Unwired Platform

Defines terms for all Sybase Unwired Platform components.

administration perspective – Or administration console. The Unwired Platform administrative perspective is the Flash-based Web application for managing Unwired Server. *See* Sybase Control Center.

administrators – Unwired Platform users to which an administration role has been assigned. A user with the "SUP Administrator" role is called a "platform administrator" and a user with the "SUP Domain Administrator" role is called a "domain administrator". These administration roles must also be assigned SCC administration roles to avoid having to authenticate to Sybase Control Center in addition to Unwired Server:

- A domain administrator only requires the "sccUserRole" role.
- A platform administrator requires both the "sccAdminRole" and "sccUserRole" roles.

Adobe Flash Player – Adobe Flash Player is required to run Sybase Control Center. Because of this player, you are required to run Sybase Control Center in a 32-bit browser. Adobe does not support 64-bit browsers.

Advantage Database Server® – A relational database management system that provides the messaging database for Sybase Unwired Platform. *See* messaging database.

Afaria® – An enterprise-grade, highly scalable device management solution with advanced capabilities to ensure that mobile data and devices are up-to-date, reliable, and secure. Afaria is a separately licensed product that can extend the Unwired Platform in a mobile enterprise. Afaria includes a server (Afaria Server), a database (Afaria Database), an administration tool (Afaria Administrator), and other runtime components, depending on the license you purchase.

application – In Unwired Server (and visible in Sybase Control Center), and application is the runtime entity that can be directly correlated to a native or mobile workflow application. The application definition on the server establishes the relationship among packages used in the application, domain that the application is deployed to, user activation method for the application, and other application specific settings.

APNS – Apple Push Notification Service.

application connection – A unique connection to the application on a device.

application connection template – a template for application connections that includes application settings, security configuration, domain details, and so forth.

application node – In Sybase Control Center, this is a registered application with a unique ID. This is the main entity that defines the behavior of device and backend interactions.

application registration – The process of registering an application with Sybase Unwired Platform. Registration requires a unique identity that defines the properties for the device and backend interaction with Unwired Server.

artifacts – Artifacts can be client-side or automatically generated files; for example: `.xml`, `.cs`, `.java`, `.cab` files.

availability – Indicates that a resource is accessible and responsive.

BAPI – Business Application Programming Interface. A BAPI is a set of interfaces to object-oriented programming methods that enable a programmer to integrate third-party software into the proprietary R/3 product from SAP®. For specific business tasks such as uploading transactional data, BAPIs are implemented and stored in the R/3 system as remote function call (RFC) modules.

BLOB – Binary Large Object. A BLOB is a collection of binary data stored as a single entity in a database management system. A BLOB may be text, images, audio, or video.

cache – The virtual tables in the Unwired Server cache database that store synchronization data.*See* cache database.

cache group – Defined in Unwired WorkSpace, MBOs are grouped and the same cache refresh policy is applied to their virtual tables (cache) in the cache database

cache partitions – Partitioning the cache divides it into segments that can be refreshed individually, which gives better system performance than refreshing the entire cache. Define cache partitions in Unwired WorkSpace by defining a partition key, which is a load argument used by the operation to load data into the cache from the enterprise information system (EIS).

cache database – Cache database. The Unwired Server cache database stores runtime metadata (for Unwired Platform components) and cache data (for MBOs). *See also* data tier.

CLI – Command line interface. CLI is the standard term for a command line tool or utility.

client application – *See* mobile application.

client object API – The client object API is described in the *Developer Guide: BlackBerry Native Applications*, *Developer Guide: iOS Native Applications*, and *Developer Guide: Windows and Windows Mobile Native Applications*.

cluster – Also known as a server farm. Typically clusters are setup as either runtime server clusters or database clusters (also known as a data tier). Clustering is a method of setting up redundant Unwired Platform components on your network in order to design a highly scalable and available system architecture.

cluster database – A data tier component that holds information pertaining to all Unwired Platform server nodes. Other databases in the Unwired Platform data tier includes the cache, messaging, and monitoring databases.

connection – Includes the configuration details and credentials required to connect to a database, Web service, or other EIS.

connection pool – A connection pool is a cache of Enterprise Information System (EIS) connections maintained by Unwired Server, so that the connections can be reused when Unwired Server receives future requests for data.

For proxy connections, a connection pool is a collection of proxy connections pooled for their respective back-ends, such as SAP Gateway.

connection profile – In Unwired WorkSpace, a connection profile includes the configuration details and credentials required to connect to an EIS.

context variable – In Unwired WorkSpace, these variables are automatically created when a developer adds reference(s) to an MBO in a mobile application. One table context variable is created for each MBO attribute. These variables allow mobile application developers to specify form fields or operation parameters to use the dynamic value of a selected record of an MBO during runtime.

data change notification (DCN) – Data change notification (DCN) allows an Enterprise Information System (EIS) to synchronize its data with the cache database through a push event.

data refresh – A data refresh synchronizes data between the cache database and a back-end EIS so that data in the cache is updated. *See also* scheduled data refresh.

data source – In Unwired WorkSpace, a data source is the persistent-storage location for the data that a mobile business object can access.

data tier – The data tier includes Unwired Server data such as cache, cluster information, and monitoring. The data tier includes the cache database (CDB), cluster, monitoring, and messaging databases.

data vault – A secure store across the platform that is provided by an SUP client.

deploy – (Unwired Server) Uploading a deployment archive or deployment unit to an Unwired Server instance. Unwired Server can then make these units accessible to users via a client application that is installed on a mobile device.

There is a one-to-one mapping between an Unwired WorkSpace project and a server package. Therefore, all MBOs that you deploy from one project to the same server are deployed to the same server package.

deployment archive – In Unwired WorkSpace, a deployment archive is created when a developer creates a package profile and executes the **build** operation. Building creates an archive that contains both a deployment unit and a corresponding descriptor file. A deployment archive can be delivered to an administrator for deployment to a production version of Unwired Server.

deployment descriptor – A deployment descriptor is an XML file that describes how a deployment unit should be deployed to Unwired Server. A deployment descriptor contains role-mapping and domain-connection information. You can deliver a deployment descriptor and a deployment unit—jointly called a deployment archive—to an administrator for deployment to a production version of Unwired Server.

deployment mode – You can set the mode in which a mobile application project or mobile deployment package is deployed to the target Unwired Server.

deployment profile – A deployment profile is a named instance of predefined server connections and role mappings that allows developers to automate deployment of multiple packages from Sybase Unwired WorkSpace to Unwired Server. Role mappings and connection mappings are transferred from the deployment profile to the deployment unit and the deployment descriptor.

deployment unit – The Unwired WorkSpace build process generates a deployment unit. It enables a mobile application to be effectively installed and used in either a preproduction or production environment. Once generated, a deployment unit allows anyone to deploy all required objects, logical roles, personalization keys, and server connection information together, without requiring access to the whole development project. You can deliver a deployment unit and a deployment descriptor—jointly called a deployment archive—to an administrator for deployment to a production version of Unwired Server.

development package – A collection of MBOs that you create in Unwired WorkSpace. You can deploy the contents of a development package on an instance of Unwired Server.

device application – *See also* mobile application. A device application is a software application that runs on a mobile device.

device notification – Replication synchronization clients receive device notifications when a data change is detected for any of the MBOs in the synchronization group to which they are subscribed. Both the change detection interval of the synchronization group and the notification threshold of the subscription determine how often replication clients receive device notifications. Administrators can use subscription templates to specify the notification threshold for a particular synchronization group.

device user – The user identity tied to a device.

DML – Data manipulation language. DML is a group of computer languages used to retrieve, insert, delete, and update data in a database.

DMZ – Demilitarized zone; also known as a perimeter network. The DMZ adds a layer of security to the local area network (LAN), where computers run behind a firewall. Hosts running in the DMZ cannot send requests directly to hosts running in the LAN.

domain administrator – A user to which the platform administrator assigns domain administration privileges for one or more domain partitions. The domain administrator has a restricted view in Sybase Control Center, and only features and domains they can manage are visible.

domains – Domains provide a logical partitioning of a hosting organization's environment, so that the organization achieves increased flexibility and granularity of control in multitenant environments. By default, the Unwired Platform installer creates a single domain named "default". However the platform administrator can also add more domains as required.

EIS – Enterprise Information System. EIS is a back-end system, such as a database.

Enterprise Explorer – In Unwired WorkSpace, Enterprise Explorer allows you to define data source and view their metadata (schema objects in case of database, BAPIs for SAP, and so on).

export – The Unwired Platform administrator can export the mobile objects, then import them to another server on the network. That server should meet the requirement needed by the exported MBO.

hostability – *See* multitenancy.

IDE – Integrated Development Environment.

JDE – BlackBerry Java Development Environment.

key performance indicator (KPI) – Used by Unwired Platform monitoring. KPIs are monitoring metrics that are made up for an object, using counters, activities, and time which jointly for the parameters that show the health of the system. KPIs can use current data or historical data.

keystore – The location in which encryption keys, digital certificates, and other credentials in either encrypted or unencrypted keystore file types are stored for Unwired Server runtime components. *See also* truststore.

LDAP – Lightweight Directory Access Protocol.

local business object – Defined in Unwired WorkSpace, local business objects are not bound to EIS data sources, so cannot be synchronized. Instead, they are objects that are used as local data store on device.

logical role – Logical roles are defined in mobile business objects, and mapped to physical roles when the deployment unit that contain the mobile business objects are deployed to Unwired Server.

matching rules – A rule that triggers a mobile workflow application. Matching rules are used by the mobile workflow email listener to identify e-mails that match the rules specified by the administrator. When emails match the rule, Unwired Server sends the e-mail as a mobile workflow to the device that matches the rule. A matching rule is configured by the administrator in Sybase Control Center.

MBO – Mobile business object. The fundamental unit of data exchange in Sybase Unwired Platform. An MBO roughly corresponds to a data set from a back-end data source. The data can come from a database query, a Web service operation, or SAP. An MBO contains both concrete implementation-level details and abstract interface-level details. At the implementation-level, an MBO contains read-only result fields that contain metadata about the data in the implementation, and parameters that are passed to the back-end data source. At the interface-level, an MBO contains attributes that map to result fields, which correspond to client properties. An MBO may have operations, which can also contain parameters that map to arguments, and which determines how the client passes information to the enterprise information system (EIS).

You can define relationships between MBOs, and link attributes and parameters in one MBO to attributes and parameters in another MBO.

MBO attribute – An MBO attribute is a field that can hold data. You can map an MBO attribute to a result field in a back-end data source; for example, a result field in a database table.

MBO binding – An MBO binding links MBO attributes and operations to a physical data source through a connection profile.

MBO operation – An MBO operation can be invoked from a client application to perform a task; for example, create, delete, or update data in the EIS.

MBO relationship – MBO relationships are analogous to links created by foreign keys in a relational database. For example, the account MBO has a field called *owner_ID* that maps to the *ID* field in the owner MBO.

Define MBO relationships to facilitate:

- Data synchronization
- EIS data-refresh policy

messaging based synchronization – A synchronization method where data is delivered asynchronously using a secure, reliable messaging protocol. This method provides fine-grained synchronization (synchronization is provided at the data level—each process communicates only with the process it depends on), and it is therefore assumed that the device is always connected and available. *See also* synchronization.

messaging database – The messaging database allows in-flight messages to be stored until they can be delivered. This database is used in a messaging based synchronization environment. The messaging database is part of the Unwired Platform data tier, along with the cache, cluster, and monitoring databases.

mobile application – A Sybase Unwired Platform mobile application is an end-to-end application, which includes the MBO definition (back-end data connection, attributes, operations, and relationships), the generated server-side code, and the client-side application code.

Mobile Application Diagram – The Mobile Application Diagram is the graphical interface to create and edit MBOs. By dragging and dropping a data source onto the Mobile Application Diagram, you can create a mobile business object and generate its attribute mappings automatically.

Mobile Application Project – A collection of MBOs and client-side, design-time artifacts that make up a mobile application.

mobile workflow packages – Mobile workflow packages use the messaging synchronization model. The mobile workflow packages are deployed to Unwired Server, and can be deployed to mobile devices, via the Unwired Platform administrative perspective in Sybase Control Center.

monitoring – Monitoring is an Unwired Platform feature available in Sybase Control Center that allows administrators to identify key areas of weakness or periods of high activity in the particular area they are monitoring. It can be used for system diagnostic or for troubleshooting. Monitored operations include replication synchronization, messaging synchronization, messaging queue, data change notification, device notification, package, user, and cache activity.

monitoring database – A database that exclusively stores data related to replication and messaging synchronization, queues status, users, data change notifications, and device notifications activities. By default, the monitoring database runs in the same data tier as the cache database, messaging database and cluster database.

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

multitenancy – The ability to host multiple tenants in one Unwired Cluster. Also known as hostability. *See also* domains.

node – A host or server computer upon which one or more runtime components have been installed.

object query – Defined in Unwired WorkSpace for an MBO and used to filter data that is downloaded to the device.

onboarding – The enterprise-level activation of an authentic device, a user, and an application entity as a combination, in Unwired Server.

operation – *See* MBO operation.

package – A package is a named container for one or more MBOs. On Unwired Server a package contains MBOs that have been deployed to this instance of the server.

palette – In Unwired WorkSpace, the palette is the graphical interface view from which you can add MBOs, local business objects, structures, relationships, attributes, and operations to the Mobile Application Diagram.

parameter – A parameter is a value that is passed to an operation/method. The operation uses the value to determine the output. When you create an MBO, you can map MBO parameters to data-source arguments. For example, if a data source looks up population based on a state abbreviation, the MBO gets the state from the user, then passes it (as a parameter/argument) to the data source to retrieve the information. Parameters can be:

- Synchronization parameters – synchronize a device application based on the value of the parameter.
- Load arguments – perform a data refresh based on the value of the argument.
- Operation parameters – MBO operations contain parameters that map to data source arguments. Operation parameters determine how the client passes information to the enterprise information system (EIS).

personalization key – A personalization key allows a mobile device user to specify attribute values that are used as parameters for selecting data from a data source. Personalization keys are also used as operation parameters. Personalization keys are set at the package level. There are three type of personalization keys: Transient, client, server.

They are most useful when they are used in multiple places within a mobile application, or in multiple mobile applications on the same server. Personalization keys may include attributes such as name, address, zip code, currency, location, customer list, and so forth.

perspective – A named tab in Sybase Control Center that contains a collection of managed resources (such as servers) and a set of views associated with those resources. The views in a perspective are chosen by users of the perspective. You can create as many perspectives as you need and customize them to monitor and manage your resources.

Perspectives allow you to group resources ways that make sense in your environment—by location, department, or project, for example.

physical role – A security provider group or role that is used to control access to Unwired Server resources.

Problems view – In Eclipse, the Problems view displays errors or warnings for the Mobile Application Project.

provisioning – The process of setting up a mobile device with required runtimes and device applications. Depending on the synchronization model used and depending on whether or not the device is also an Afaria client, the files and data required to provision the device varies.

pull synchronization – Pull synchronization is initiated by a remote client to synchronize the local database with the cache database. On Windows Mobile, pull synchronization is supported only in replication applications.

push synchronization – Push is the server-initiated process of downloading data from Unwired Server to a remote client, at defined intervals, or based upon the occurrence of an event.

queue – In-flight messages for a messaging application are saved in a queue. A queue is a list of pending activities. The server then sends messages to specific destinations in the order that they appear in the queue. The depth of the queue indicates how many messages are waiting to be delivered.

relationship – *See* MBO relationship.

relay server – *See also* Sybase Hosted Relay Service.

resource – A unique Sybase product component (such as a server) or a subcomponent.

REST web services – Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

RFC – Remote Function Call. You can use the RFC interface to write applications that communicate with SAP R/3 applications and databases. An RFC is a standalone function. Developers use SAP tools to write the Advanced Business Application Programming (ABAP) code that implements the logic of a function, and then mark it as "remotely callable," which turns an ABAP function into an RFC.

role – Roles control access to Sybase Unwired Platform resources. *See also* logical role and physical role.

role mapping – Maps a physical (server role) to a logical (Unwired Platform role). Role mappings can be defined by developers, when they deploy an MBO package to a development Unwired Server, or by platform or domain administrators when they assign a security configuration to a domain or deploy a package to a production Unwired Server (and thereby override the domain-wide settings in the security configuration).

RSOE – Relay Server Outbound Enabler. An RSOE is an application that manages communication between Unwired Server and a relay server.

runtime server – An instance of Unwired Server that is running. Typically, a reference to the runtime server implies a connection to it.

SAP – SAP is one of the EIS types that Unwired Platform supports.

SCC – Sybase Control Center. A Web-based interface that allows you to administer your installed Sybase products.

schedule – The definition of a task (such as the collection of a set of statistics) and the time interval at which the task must execute in Sybase Control Center.

scheduled data refresh – Data is updated in the cache database from a back-end EIS, based on a scheduled data refresh. Typically, data is retrieved from an EIS (for example, SAP) when a device user synchronizes. However, if an administrator wants the data to be preloaded for a mobile business object, a data refresh can be scheduled so that data is saved locally in a cache. By preloading data with a scheduled refresh, the data is available in the information server when a user synchronizes data from a device. Scheduled data refresh requires that an administrator define a cache group as "scheduled" (as opposed to "on-demand").

security configuration – Part of the application user and administration user security. A security configuration determines the scope of user identity, authentication and authorization checks, and can be assigned to one or more domains by the platform administrator in Sybase Control Center. A security configuration contains:

- A set of configured security providers (for example LDAP) to which authentication, authorization, attribution is delegated.
- Role mappings (which can be specified at the domain or package level)

security provider – A security provider and it's repository holds information about the users, security roles, security policies, and credentials used by some to provide security services to Unwired Platform. A security provider is part of a security configuration.

security profile – Part of the Unwired Server runtime component security. A security profile includes encryption metadata to capture certificate alias and the type of authentication used by server components. By using a security profile, the administrator creates a secured port over which components communicate.

server connection – The connection between Unwired WorkSpace and a back-end EIS is called a server connection.

server farm – *See also* cluster. Is the relay server designation for a cluster.

server-initiated synchronization – *See* push synchronization.

SOAP – Simple Object Access Protocol. SOAP is an XML-based protocol that enables applications to exchange information over HTTP. SOAP is used when Unwired Server communicates with a Web service.

solution – In Visual Studio, a solution is the high-level local workspace that contains the projects users create.

Solution Explorer – In Visual Studio, the Solution Explorer pane displays the active projects in a tree view.

SSO – Single sign-on. SSO is a credential-based authentication mechanism.

statistics – In Unwired Platform, the information collected by the monitoring database to determine if your system is running as efficiently as possible. Statistics can be current or historical. Current or historical data can be used to determine system availability or performance. Performance statistics are known as key performance indicators (KPI).

Start Page – In Visual Studio, the Start Page is the first page that displays when you launch the application.

structured data – Structured data can be displayed in a table with columns and labels.

structure object – Defined in Unwired WorkSpace, structures hold complex datatypes, for example, a table input to a SAP operation.

subscription – A subscription defines how data is transferred between a user's mobile device and Unwired Server. Subscriptions are used to notify a device user of data changes, then these updates are pushed to the user's mobile device.

Sybase Control Center – Sybase Control Center is the Flash-based Web application that includes a management framework for multiple Sybase server products, including Unwired Platform. Using the Unwired Platform administration perspective in Sybase Control Center, you can register clusters to manage Unwired Server, manage domains, security configurations, users, devices, connections, as well as monitor the environment. You can also deploy and MBO or workflow packages, as well as register applications and define templates for them. Only use the features and documentation for Unwired Platform. Default features and documentation in Sybase Control Center do not always apply to the Unwired Platform use case.

Sybase Control Center *X.X* Service – Provides runtime services to manage, monitor, and control distributed Sybase resources. The service must be running for Sybase Control Center to run. Previously called Sybase Unified Agent.

Sybase Hosted Relay Service – The Sybase Hosted Relay Service is a Web-hosted relay server that enables you to test your Unwired Platform development system.

Sybase Messaging Service – The synchronization service that facilitates communication with device client applications.

Sybase Unwired Platform – Sybase Unwired Platform is a development and administrative platform that enables you to mobilize your enterprise. With Unwired Platform, you can develop mobile business objects in the Unwired WorkSpace development environment, connect to structured and unstructured data sources, develop mobile applications, deploy mobile business objects and applications to Unwired Server, which manages messaging and data services between your data sources and your mobile devices.

Sybase Unwired WorkSpace – Sybase Unwired Platform includes Unwired WorkSpace, which is a development tool for creating mobile business objects and mobile applications.

synchronization – A synchronization method where data is delivered synchronously using an upload/download pattern. For push-enabled clients, synchronization uses a "poke-pull" model, where a notification is pushed to the device (poke), and the device fetches the content (pull), and is assumed that the device is not always connected to the network and can operate in a disconnected mode and still be productive. For clients that are not push-enabled, the default synchronization model is pull.*See also* messaging based synchronization.

synchronization group – Defined in Unwired WorkSpace, a synchronization group is a collection of MBOs that are synchronized at the same time.

synchronization parameter – A synchronization parameter is an MBO attribute used to filter and synchronize data between a mobile device and Unwired Server.

synchronization phase – For replication based synchronization packages, the phase can be an upload event (from device to the Unwired Server cache database) or download event (from the cache database to the device).

synchronize – *See also* data refresh. Synchronization is the process by which data consistency and population is achieved between remote disconnected clients and Unwired Server.

truststore – The location in which certificate authority (CA) signing certificates are stored. *See also* keystore.

undeploy – Running **undeploy** removes a domain package from an Unwired Server.

Unwired Server – The application server included with the Sybase Unwired Platform product that manages mobile applications, back-end EIS synchronization, communication, security, transactions, and scheduling.

user – Sybase Control Center displays the mobile-device users who are registered with the server.

view – A window in a perspective that displays information about one or more managed resources. Some views also let you interact with managed resources or with Sybase Control Center itself. For example, the Perspective Resources view lists all the resources managed by the current perspective. Other views allow you to configure alerts, view the topology of a replication environment, and graph performance statistics.

Visual Studio – Microsoft Visual Studio is an integrated development environment product that you can use to develop device applications from generated Unwired WorkSpace code.

Welcome page – In Eclipse, the first set of pages that display when you launch the application.

workspace – In Eclipse, a workspace is the directory on your local machine where Eclipse stores the projects that you create.

WorkSpace Navigator – In Eclipse, the tree view that displays your mobile application projects.

WSDL file – Web Service Definition Language file. The file that describes the Web service interface that allows clients to communicate with the Web service. When you create a Web service connection for a mobile business object, you enter the location of a WSDL file in the URL.

Index