System Administration

SAP Mobile Platform 2.3

DOCUMENT ID: DC01931-01-0230-02
LAST REVISED: July 2013

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

Contents

CHAPTER 1 Documentation Roadmap for SAP Mobile Platform

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

See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role.

Check the Product Documentation Web site regularly for updates: *http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories*, then navigate to the most current version.

CHAPTER 2 Administration of the SAP Mobile Platform

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

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

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

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

  Note: The domain administrator for the "default" domain has the same level of privileges as the SAP Mobile Platform administrator.
- Helpdesk – has read-only access to all administration information in the SAP Mobile Platform administration console. In a deployment edition, you must map the SUP Helpdesk role to a role in your existing repository.

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

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

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

Before you begin, know: the devices you need to support, the back-ends you need to integrate with, and the synchronization model you will use.

- Mobile application administration requires both the development and deployment of applications. The SAP Mobile Platform perspective in SAP Control Center is integral to configuring and managing applications as they are developed and integrated into your system landscape. Further, all aspects of a production environment can be monitored for troubleshooting or performance tuning purposes.

  Before you begin, decide: the system design/topology you environment requires and what type of security providers you need to delegate application security to.
- Mobile user, device, and application management simplifies how the end user is registered and provisioned in the SAP Mobile Platform environment. When Afaria® is used in conjunction with SAP Mobile Platform, an administrator has a powerful cross-platform mobile management framework:
  - SAP Control Center performs the package deployment to the SAP Mobile Server as well as manages user accounts and data service subscriptions.

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

SAP Mobile Platform Security

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

Security consists of:

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

Security is extensively documented in the *Security* guide.

Administration Areas Not Applicable to Online Data Proxy

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

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

In general, the following areas do not apply:

- Synchronization / Data Change Notification (DCN)
- SNMP Notifications

- Replication protocol
- Mobile Business Objects (MBOs), MBO packages
- Mobile Workflows or Hybrid Web Containers
- Package Administration
- Status and Performance Monitoring

Administering Agentry Applications

Agentry applications are published from Agentry Editor and deployed to SAP Mobile Server. Each Agentry application has its own Agentry Server instance that runs on the SAP Mobile Server node. The Agentry Server instance synchronizes all production data between the back-end system and the Agentry Clients, including system connections, client-server communications, and authentication.

You can modify the configuration of the Agentry application server instance in SAP Control Center, based on the information loaded from the Agentry Server configuration file. Some configuration files must be updated directly.

Using SAP Control Center you can also:

- Start, stop, or restart an Agentry Server.
- View information about users currently connected to the Agentry Server, and disconnect those users from the Server.
- View information about the Agentry Server and the operating system on which the Agentry Server is running.
- Enable, disable, and set the verbosity level of Agentry server logs.
- Roll and view Agentry application message and event log files generated by the Agentry Server.

All other administration functions performed in SAP Control Center do not apply to Agentry applications.

CHAPTER 3 Cluster Administration

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

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

Table 1. Cluster administration tasks

Task	Frequency	Accomplished by
Installing the cluster	One-time installation per cluster	SAP Mobile Platform installer
Setting up Relay Servers	One-time initial installation and configuration; occasionally adding servers to the cluster	Manual installation; manual setup using configuration files.
Suspending and resuming server nodes	On demand, as required	SAP Control Center
Setting cluster properties, including cache database settings, monitoring database setup, and so on	Once, or as cluster changes require	Manual configuration using files and `.BAT` scripts.

Task	Frequency	Accomplished by
Configuring the cluster to: • Set the configuration cache properties • Set the solution manager URL • Set the replication ports and properties • Set the messaging synchronization ports • Set the amount of data that can be downloaded through an HTTP connection using the BlackBerry MDS • Set the management ports for communication requests from SAP Control Center • Configure the client dispatcher • Set how applications handle DCN requests sent though an HTTP GET operation • Create security profiles for secure communication • Set up secure synchronization • Tune server performance	Post installation configuration with infrequent tuning as required	See *SAP Control Center for SAP Mobile Platform>Administer> Clusters*
Setting cluster log file settings for SAP Mobile Server system components	Once, unless log data requirements change	See *SAP Control Center for SAP Mobile Platform>Administer>Clusters*
Administering the runtime databases	Routine to ensure that the database server is monitored and backed up, that there is sufficient space for SAP Mobile Platform metadata and cached data tables, and that performance is within acceptable limits (performance tuning)	Established processes and command line utilities. Consult with your database administrator.

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

Configuring Asynchronous Operation Replay Queue Count

Configure properties of a cluster to control whether asynchronous operation replays are enabled for all cluster packages.

1. In left navigation pane of SAP Control Center, click the name of the cluster.
2. In the right administration pane, click **General**.
3. Configure the queue limit for asynchronous operation replays in **Asynchronous operation replay queue count**. The minimum acceptable queue count is 1 and the default is 5.
4. Click **Save**.

Configuring Cluster Performance Properties

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

1. In the left navigation pane, select **Configuration**.
2. In the right administration pane, click the **General** tab.
3. From the menu bar, select **Performance**.
4. Configure these properties, as required:

   - Inbound messaging queue count – number of message queues used for incoming messages from the messaging-based synchronization application to the server. SAP recommends you choose a value that represents at least 10% of active devices.
   - Maximum count of total webservice connections – maximum number of web service connections allowed overall.
   - Maximum count of webservice connections per host – maximum number of web service connections allowed per host configuration.
   - Maximum number of in memory messages – maximum allowable number of in memory messages.
   - Maximum proxy connection pool size – maximum size for the replication protocol server memory cache.
   - Outbound messaging queue count – number of message queues used for outbound messages from the server to the messaging-based synchronization application. SAP

recommends a value that represents at least 50% of active devices. However, if you are running 32-bit operating system, do not exceed a value of 100% of active devices.

- Subscribe bulk load thread pool size – maximum number of threads allocated to initial bulk load subscription operations. The default value is 5 Setting the thread pool size too high can impact performance.
- Synchronization cache size – maximum size for the replication protocol server memory cache.
- Thread count – MobiLink thread count. This value should be lower than the thread count for the SQL Anywhere database.
- Webservice connection timeout – length of time, in seconds, until a web service connection is established. A value of 0 (zero) means no timeout.

5. Click **Save**.

Replication

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

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

See *Replication* topics in *SAP Control Center for SAP Mobile Platform*.

Configuring a Replication Listener

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

Prerequisites

A secure synchronization stream uses SSL or TLS encryption. Both TLS and SSL require production-ready certificates to replace the default ones installed with SAP Mobile Server. Ensure that you possess digital certificates verified and signed by third-party trusted authorities. See *Encrypting Synchronization for Replication Payloads* in *Security*.

Task

1. Open SAP Control Center.

2. In the left navigation pane, select **Configuration** .
3. In the right administration pane, click the **General** tab.
4. From the menu bar, select **Components**.
5. Select **Replication** and click **Properties**.
6. Select the protocol and port you require:
   - If you do not require SSL encryption, choose **Port**. SAP Mobile Platform recommends this option if you do not require a secure communication stream for synchronization. By default, the port for HTTP is 2480.
   - To encrypt the HTTP stream with SSL, choose **Secure port**. By default, the port for HTTPS is 2481. The "Secure Sync Port" properties can be used to review and set the server identity and public certificate for the secure synchronization port. See below.
7. (Optional) Configure additional properties for E2EE with TLS, HTTPS with SSL, and synchronization server startup options:

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

   - E2E Encryption Certificate – specify the file containing the private key that acts as the identity file for SAP Mobile Server.
   - E2E Encryption Certificate Password – set the password to unlock the encryption certificate.
   - E2E Encryption Public Key – specify the file containing the public key for SAP Mobile Server.
   - E2E Encryption Type – specify the asymmetric cipher used for key exchange for end-to-end encryption. You can only use RSA encryption.
   - Secure Sync Port Certificate – identifies the location of the security certificate used to encrypt and decrypt data transferred using SSL.
   - Secure Sync Port Certificate Password – is used to decrypt the private certificate listed in certificate file. You specify this password when you create the server certificate for SSL.
   - Secure Sync Port Public Certificate – specify the file containing the SSL public key that acts as the identity file for synchronization port.
   - Trusted Relay Server Certificate – if Relay Server trusted certificate is configured for HTTPS connections encrypted with SSL, identifies the public security certificate location.
   - User Options – sets the command line options for starting the synchronization server. These options are appended the next time the synchronization server starts. These are the available user options:

Option	Description
`@ [variable | filePath]`	Applies listener options from the specified environment variable or text file.

Option	Description
`-a <value>`	Specifies a single library option for a listening library.
`-d <filePath>`	Specifies a listening library.
`-e <deviceName>`	Specifies the device name.
`-f <string>`	Specifies extra information about the device.
`-gi <seconds>`	Specifies the IP tracker polling interval.
`-i <seconds>`	Specifies the polling interval for SMTP connections.
`-l <"keyword=value;...">`	Defines and creates a message handler.
`-m`	Turns on message logging.
`-ni`	Disables IP tracking.
`-ns`	Disables SMS listening.
`-nu`	Disables UDP listening.
`-o <filePath>`	Logs output to a file. **Note:** Ensure that you enter the absolute file path for this property.
`-os <bytes>`	Specifies the maximum size of the log file.
`-p`	Allows the device to shut down automatically when idle.
`-pc [+ | -]`	Enables or disables persistent connections.
`-r <filePath>`	Identifies a remote database involved in the responding action of a message filter.
`-sv <scriptVersion>`	Specifies a script version used for authentication.
`-zf`	(Recommended in development environments) Causes the SAP Mobile Server replication service to check for script changes at the beginning of each synchronization. Unless this option is used, the service assumes that no script changes have been made, no checks for script changes are performed, once the service starts. For production environments, this option is not recommended due to its negative impact on synchronization performance.

Option	Description
`-t [+ | -] <name>`	Registers or unregisters the remote ID for a remote database.
`-u <userName>`	Specifies a synchronization server user name.
`-v [0 | 1 | 2 | 3]`	Specifies the verbosity level for the messaging log.
`-y <newPassword>`	Specifies a new synchronization server password.

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

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

8. Click **OK**.

Messaging

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

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

See *Messaging* topics in *SAP Control Center for SAP Mobile Platform*.

Configuring Messaging Properties

Configure a port to receive service requests from devices.

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

   There are limits on the amount of data that can be downloaded through an HTTP connection using the BlackBerry MDS. The BES response portioning limit determines the amount of HTTP traffic the BES MDS server accepts from SAP Mobile Server. For

BlackBerry MDS, this limit is set in BlackBerry Manager using the Maximum KB/ Connection setting.

8. Click **OK**.

Cluster License Coordination

In a cluster, each server deployed to the environment must be licensed. Multiple servers cannot share a single license. However, all server nodes in the cluster can share device connection licenses.

In a clustered environment, you must use a license server so it can coordinate licensing requirements among all installed components:

- Server validation – each time a server starts, it connects and registers with the license server to check if there is a valid license for it. If there is a free license available, the server checks out the license and continues with the start-up process. If the number of licensed servers cannot be retrieved from the license file, or the license server confirms that a server is not licensed, SAP Mobile Server stops.
- Device connection validation – because available device licenses are shared among all servers in the cluster, all connections to all servers must be accounted for. The cluster name enumerates each device connection made across clustered servers. Every server then checks out all device licenses when the servers start.

Checking System Licensing Information

Review licensing information to monitor used mobile device licenses, license expiry dates, and other license details. This information allows administrators to manage license use and determine whether old or unused device licenses should be transferred to new devices.

1. In the left navigation pane, select the top-level tree node.
2. In the right administration pane, select the **General** tab, and click **Licensing**.
3. Review the following licensing information:
   - Server license type – the type of license currently used by SAP Mobile Platform. For more information on license types, see *SAP Mobile Platform License Types*.
   - Production edition – the edition of the software you have installed.
   - Server license expiry date – the date and time at which the server license expires. When a server license expires, SAP Mobile Server generates a license expired error and SAP Mobile Server is stopped.
   - Overdraft mode – allows you to generate additional licenses in excess of the quantity of licenses you actually purchased. This enables you to exceed your purchased quantity of licenses in a peak usage period without impacting your operation. This mode is either enabled or disabled, as specified by the terms of the agreement presented when you obtain such a license.

- Used mobile user license count – the number of mobile user licenses currently in use. A mobile user is a distinct user identity—username and associated security configuration—that is registered in the server. As such, the used mobile user license count represents the total distinct user identities registered on the server. One mobile user may access:
  - Multiple applications and different versions of the same application.
  - The same or different versions of an application from multiple devices.

4. Click **Close**.

Note: SAP Mobile Platform licensing is configured during installation. However, if necessary, license details can be changed at a later time.

CHAPTER 4 Server Administration

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

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

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

Table 2. Server administration tasks

Task	Frequency	Accomplished by
Installing the server	One-time installation per server	SAP Mobile Platform installer.
Tuning the server performance.	Post installation with infrequent tuning as required	SAP Control Center. See *SAP Control Center for SAP Mobile Platform>Administer> SAP Mobile Server*
Manage the outbound enabler configuration for Relay Server. • Configure Relay Server properties • Manage certificates • View logs • Configure proxy servers for outbound enabler	Post installation	SAP Control Center

Changing SAP Mobile Server Host Name

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

1. Stop the SAP Mobile Server service.
2. Update all the `socketlistener` property files related to the node with the changed host name. Change the value of host to the new host name. The property files are located at *SMP_HOME*`\Servers\UnwiredServer\Repository\Instance\com\sybase\djc\server\SocketListener`

\<*mlservername*>_<*protocol*>.properties. Open each file, and change the old host value name to the new host name.

3. Update the property value in the cluster database:
   a) Using dbisqlc, connect to the cluster database.
   b) Update the sup.host property value with the new node value. For example:
      `update MEMBER_PROP set value1='<new_hostname>' where name='sup.host' and value1='<old_hostname>'`
4. Update the property value of sup.host with the new host name in `sup.properties`. It is located at: `Repository\Instance\com\sybase\sup\server\SUPServer\sup.properties`.
5. To enable SAP Control Center to work:
   a) Restart the SAP Control Center *X.X* service.
   b) Modify the URL in the shortcut for the SAP Control Center. Use the new hostname instead of the old one in the launched URL
6. For a cluster environment, repeat step 2 for each node.
7. Restart the SAP Mobile Server service.

Starting and Stopping SAP Mobile Server

You can start and stop SAP Mobile Server in different ways, depending on the use context.

Review this table to understand which method you should use.

Method	Use When	Services Started or Stopped
SAP Control Center SAP Mobile Server list	Stopping or starting remote SAP Mobile Server nodes	SAP Mobile Server service only
Desktop shortcut	Stopping or starting SAP Mobile Server locally	All runtime services installed on that host
Windows Services panel	Stopping or starting SAP Mobile Server locally	Any combination of individual services that require stopping

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

Stopping and Starting a Server from SAP Control Center

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

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

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

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

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

1. In the SAP Control Center navigation pane, click **Servers** to display the servers list.
2. Select a server in this list.
3. Choose an appropriate process:

   - To stop the server, click **Stop**. You can then perform the administration actions you require that might require the server to be started. To then restart the server, click **Start**.

     Note: If you have selected a Scale-out node server type, the **Start** button is disabled.
   - If you perform an administration action that requires a restart to take effect, click **Restart**. This shuts the server down and restarts it in a single process.

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

Stopping and Starting a Server from the Desktop Shortcut

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

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

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

Configuring SAP Mobile Server General Properties

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

1. Open SAP Control Center.
2. In the left navigation pane, expand the **Servers** folder and select a server.
3. Select **Server Configuration**.
4. In the right administration pane, select the **General** tab.
5. Configure these replication payload properties, as required:

   - Host Name – the name of the machine where SAP Mobile Server is running (read only).
   - Maximum Heap Size – the maximum size of the JVM memory allocation pool. Use K to indicate kilobytes, M to indicate megabytes, or G to indicate gigabytes. For production recommendations on this value, see *SAP Mobile Server Replication Tuning Reference* in *System Administration*.
   - Minimum Heap Size – the minimum size of the JVM memory allocation pool, in megabytes. For production recommendations on this value, see *SAP Mobile Server Replication Tuning Reference* in *System Administration*.
   - Thread Stack Size – the JVM `-Xss` option.
   - User Options (in Show optional properties) – other JVM options.

     For example, you can enable JVM garbage collection logging by setting `-XX:+PrintGCDetails`. Or you can set the permanent space which is allocated outside of the Java heap with the `DJC_JVM_MAXPERM` environment variable (in *`SMP_HOME`*`\Servers\UnwiredServer\bin\usersetenv.bat`); the maximum perm size must be followed by K, M, or G, for example, `-XX:MaxPermSize=512M`. Note that DJC_JVM_MAXPERM is not visible to SAP Control Center.

     If you are in an SAP environment and want to define the Solution Manager URL into which trace files can be uploaded, use this user option: `-Dcom.sap.solutionmanager.url=<SolutionManager_URL>`.

     Note: Enter options as a series of Java system property and value pairs, delimited by spaces, with this syntax:

     ```
     -DpropertyName1=value1 -DpropertyName2=value2
     ```

     For information on troubleshooting JVM settings, see *SAP Mobile Server Fails to Start Due to Incorrect JVM Settings* in *Troubleshooting*.
6. Click **Save**.

See also

- *Tuning Synchronization for Messaging Payloads* on page 102
- *Testing CPU Loads for Replication* on page 92

Configuring SAP Mobile Server to Prepare for Connections Outside the Firewall

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

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

Relay Server Components Used in SAP Mobile Platform

Relay Server operations include several executable components.

These components include:

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

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

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

Relay Server Outbound Enabler

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

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

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

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

See *Relay Server Outbound Enabler* topics in *SAP Control Center for SAP Mobile Platform*.

CHAPTER 5 Agentry Server Administration

Each Agentry application has its own Agentry Server instance that runs on the SAP Mobile Server node. Each Agentry application server instance is initially configured based on the information loaded from the primary Agentry Server configuration file, `Agentry.ini`, which is installed with SAP Mobile Server.You can modify the configuration for individual Agentry applications using SAP Control Center. To modify the `Agentry.ini` file installed with the SAP Mobile Server, you must edit the file directly.

Note: The `Agentry.ini` file installed with SAP Mobile Server does not contain any settings in the System Connections section, which is used for configuring backend communications. System connections are defined in the application project and after the application file is deployed, these settings can be modified using SAP Control Center. Some modifications may require editing additional configuration files.

For details about the contents of the `Agentry.ini` file, see *Agentry Configuration Files*. For more information about configuring Agentry application server instances using SAP Control Center, see *Creating Agentry Application Definitions* in *SAP Control Center for SAP Mobile Platform*.

Note: In a clustered SAP Mobile Platform system, if you need to modify Agentry configuration files outside of SAP Control Center, you cannot modify the files in the SAP Mobile Server directory tree structure. Instead, modify the files outside this structure and add them to a zip file that can be deployed using SAP Control Center. Deploy the zip file to the Agentry application on the primary SAP Mobile Server. The configuration changes will then be propagated to all other nodes in the cluster, as appropriate.

Agentry Server Configuration Overview

Once the Agentry Server has been installed, it will require configuration specific to the implementation environment. This configuration affects all areas of the Server's behavior, including:

- Agentry Server-back end communications
- Client-Agentry Server communications
- Agentry Server logging and log file management
- Security and Authentication
- General Agentry Server behavior

Configuring the Agentry Server involves changes to the various configuration files. Which files are modified depends on the areas of behavior and functionality in need of configuration. The primary configuration file for the Agentry Server is the file `Agentry.ini`. However,

any of several other files may need to be modified as well. Much of the initial configuration, including settings in the `Agentry.ini` file, as well as log file settings can be set using the SAP Control Center. Other configuration settings are modified directly in the Server's configuration files.

For Windows installations, all configuration files for the Agentry Server are located in directory within the SAP Mobile Platform installation location for the Agentry Server instance for a specific Agentry application (e.g. `C:\SAP\MobilePlatform\Servers\UnwiredServer\Repository\Agentry\default\<AppName>`).

When modifying the configuration files directly, it is highly recommended that they first be copied to a back-up location. When modifying the files you must always use a standard text editor such as Notepad. The SAP Control Center will make a back up copy of the `Agentry.ini` and the `Logs.ini` files before applying changes to them.

Agentry Server - Back End Communications Overview

The communications between the Agentry Server and the back end system are represented within the application by the system connection definition type. Within the `Agentry.ini` configuration file for the Agentry Server, each system connection has a corresponding section. Within this section are the configuration options specific to the type of back-end system. The settings for these options are then dependent on the specific system to which the Server will connect. The four types of sections within the file are:

- `[SQL-n]` - For SQL database system connections (Both MS SQL Server and Oracle)
- `[Java-n]` - For Java Virtual Machine system connections
- `[HTTPXML-n]` - For HTTP-XML Server (a.k.a. Web Service) system connections
- `[File-n]` - For connections to the operating system upon which the Server is installed

Another section controlling these communications is `[System Connections]`. Listed in this section needs to be one entry for each back end system with which the Agentry Server will connect. The specific entry within this section for a system connection consists of the system connection definition's ID (assigned automatically and displayed in the Agentry Editor) and the data link library, or DLL file for that system connection type. This will be one of the following values:

- `ag3sqlbe.dll` - For SQL database system connections
- `ag3javabe.dll` - For Java Virtual Machine system connections
- `ag3httpxmlbe.dll` - For HTTP-XML Server system connections
- `ag3filebe.dll` - For connections to the Windows system upon which the Server is installed.

Tying each of these three components for a system connection together is the unique numeric identifier generated by the Agentry Editor for the system connection definition. The definition itself is referenced internally by this identifier at run-time. The entries in the `[System Connections]` section are also identified by this value, as in `1=ag3sqlbe.dll`. Finally,

the names of the configuration sections for a system connection within the `Agentry.ini` file also include this identifier, as in `[SQL-1]`. The numeric value `1` in each of the preceding examples must also be the same numeric identifier for the SQL Database system connection in the Editor.

By default the `Agentry.ini` file for a newly installed Agentry Server contains no settings for a system connection. The `[System Connections]` section exists but is empty. The connection-specific sections (`[SQL-1]`, `[Java-2]`, etc.) are not included in the default `Agentry.ini` file. These sections and their configuration options must be added to the file as a part of the configuration process. To set these configuration options the Agentry Editor and the SAP Control Center are used. The overall process is as follows:

1. After the application project is in a state suitable for publishing, perform a publish from the Agentry Editor. The Agentry Editor will prompt to create the necessary configuration sections in the `Agentry.ini` file. This will be those related to all system connections defined in the application project. The sections will be created with initial settings likely in need of further modification after the publish is complete.
2. After the initial publish, the SAP Control Center is used to set the final settings for each system connection, as well as the client-server communications. This includes changes to the sections for each system connection and the ANGEL (Agentry Next Generation Encryption Layer) settings, that dictate the behavior of the Client-Server communications.
3. Setting the options related to log files and log message categories is also performed using the SAP Control Center. The specifics of these settings will depend on the type of Agentry Server, Development or Production, being configured and the overall log needs of the system. Many of these log categories are related to the Agentry Server-back end communications, as well as other log messages.

SQL Database System Connection Configuration Overview

The following tasks are performed when configuring the Agentry Server to communicate over a SQL Database system connection. Certain tasks may or may not pertain to a particular implementation environment and a careful review of the application's requirements and functionality should be performed before proceeding:

1. Research and information gathering on the database with which the Agentry Server is to connect, including:
   - Database identifiers (e.g. ODBC System DSN names or Oracle Local Net Service Configuration Names)
   - Database login information for use by the Agentry Server
   - Back end user authentication needs of the application
   - Other relevant communications information for the back end system.
2. Modifications to the Agentry Server's `Agentry.ini` file, specifically the sections:
   - `[System Connections]`
   - `[SQL-n]`

These modifications should be made using the SAP Control Center, rather than manually modifying the configuration file.

3. Application-specific modifications to the Agentry Server's `SqlBe.ini` configuration file.
4. Application-specific additions or modifications to the query constants file, or the creation of additional constants files.

Java Virtual Machine System Connection Configuration Overview

The following tasks are related to configuring the Agentry Server to communicate over a JVM system connection. Certain tasks may or may not pertain to a particular implementation environment and a careful review of the application's requirements and functionality should be performed before proceeding:

1. Research and information gathering on the Java interface with which the Agentry Server is to connect, including:
   - Host system network name of the Java application server with which the Agentry Server is to connect.
   - Application server name of the Java interface, where applicable.
   - Back end user authentication needs of the application.
   - Other relevant communications information for the back end system.
2. Modifications to the Agentry Server's `Agentry.ini` file, specifically the sections:
   - `[System Connections]`
   - `[Java-n]`

   These modifications should be made using the SAP Control Center, rather than manually modifying the configuration file.

HTTP-XML Server System Connection Configuration Overview

The following tasks are involved in configuring the Agentry Server to communicate over an HTTP-XML system connection. Certain tasks may or may not pertain to a particular implementation environment and a careful review of the application's requirements and functionality should be performed before proceeding:

1. Research and information gathering for the web server with which the Agentry Server is to connect, including:
   - The base URL address with which the Agentry Server is to connect.
   - If specific name spaces exist in the XML schema to be used, those names should be noted.
   - Back end user authentication needs of the application
   - Other relevant communications information for the back end system, including port numbers and network timeout durations.
2. Modifications to the Agentry Server's `Agentry.ini` file, specifically the sections:

- `[System Connections]`
- `[HTTPXML-n]`

These modifications should be made using the SAP Control Center, rather than manually modifying the configuration file.

3. Application-specific additions or modifications to the XML constants file, or the creation of additional constants files.

File System Connection Configuration Overview

The following tasks are involved in configuring the Agentry Server to communicate over a File system connection. Certain tasks may or may not pertain to a particular implementation environment and a careful review of the application's requirements and functionality should be performed before proceeding:

1. Research and information gathering on the host system upon which the Agentry Server is to running, including:
   - User permissions required to perform the needed tasks of the application.
   - User login information for the Agentry Server.
   - Back end user authentication needs of the application.
2. Create a user account for the Agentry Server to connect to the host system.
3. Modifications to the Agentry Server's `Agentry.ini` file, specifically the sections:
   - `[System Connections]`
   - `[File-n]`

   These modifications should be made using the SAP Control Center, rather than manually modifying the configuration file.

Client-Server Communications Overview

The behavior of client-server communications within Agentry are the result of the transmit configuration definitions within the application and the corresponding configuration settings of the Agentry Server. The configuration of the Agentry Server involves the modification of the `Agentry.ini` file and/or making changes through the SAP Control Center. There are several sections within this configuration file and which is modified is dependent on the type of communications in use for an implementation. The sections below correspond to the connection type of the transmit configuration definitions within the application. The default and recommended connection type is the Agentry Next Generation Encryption Layer protocol, or ANGEL.

Following are the sections that may need to be modified:

- `[Front Ends]` - This is a list of the proper Dynamic Link Libraries (DLL's) needed for the connection types used within the application. This is dictated by the transmit configuration definitions within the application.

- `[ANGEL Front End]` - This is the configuration section for connections using the Agentry Next Generation Encryption Layer protocol. This is the default connect type for any Agentry application project and new transmit configurations.
- `[ANGEL Front End Ports]` - This section contains a list of the domains and port numbers upon which the Agentry Server will listen for Agentry Client requests made using the ANGEL protocol.
- `[Web Server Front End]` - This is the configuration section for the Web Client. This is used when users access the application via Internet Explorer or the Blackberry Handheld Browser.
- `[Web Server MIME Types]` - This section lists the MIME types for files returned to the web browsers by the Web Server Front End. The items listed here will be added to the MIME types contained in the Windows registry. If an overlap occurs, the settings listed in this section will override the MIME type settings of the host system.

ANGEL Communications Protocol Overview

The Agentry Next Generation Encryption Layer (ANGEL) communications protocol is the default connect type in Agentry versions 4.4 and later. This connection type provides secure communications using SSL/TLS encryption over TCP/IP connections. Previous releases of Agentry contained a Secure Network Connection protocol. ANGEL replaces this connection and is the recommended protocol for all transmit configurations within an application deployment. The `Agentry.ini` sections `[ANGEL Front End]` and `[ANGEL Front End Ports]` control the Server's behavior for connections made using a transmit configuration with a connect type of ANGEL.

Unlike other connection types in Agentry, and in addition to its encryption behavior, Clients connecting using the ANGEL protocol open a single socket for communications with the Server. Also unlike other connection types, the Server is capable of receiving requests to multiple domains and listening on multiple ports for Client requests.

The final reason ANGEL is both the default connection type and is the one recommended for use is that the Midstation and Unsecured Network Connection types currently available for a transmit configuration definition are scheduled to be deprecated in a future release of Agentry.

Web Browser/BlackBerry® Handheld Browser Clients

The Web Browser (Internet Explorer) and BlackBerry® Handheld Browser (RIM BlackBerry devices) clients, collectively referred to here as "browser clients," allow for applications to be deployed to users as HTML or WML pages. The Agentry Server in this case acts as a web server, serving up pages to the browser client that contain the UI of the application. The application must have the proper platform and screen definitions within its screen sets to support this behavior. Unlike the other connection types, the browser client does not correspond to a transmit configuration connection type.

The two configuration sections involved in configuring these communications are `[Web Server Front End]` and `[Web Server MIME Types]`. This connection type uses SSL to encrypt communications. This can also be authenticated SSL. The Server's

authentication certificates used by the `[ANGEL]` communications may also be used here. The web browsers on the clients must have the Server's authentication certificate as a trusted certificate in order for synchronization to occur.

There are several aspects to the behavior of the application when the browser client's are in use: all "client-side" processing is performed by the Server; certain functionality is implemented differently, or is not supported by the browser client; and other items. Differences are discussed in detail in the *Agentry Reference Guide*.

Transmit Configuration Definitions

The `TransmitConfigurationsBase.ini` file contains one section for each transmit configuration defined within the application, denoted as `[`*`TransmitConfiguratioName`*`]`. The items within a section correspond to all of the attributes within the definition, with the values being those defined for the transmit configuration. The override file values will override those attributes on the Clients for the transmit configuration.

This file is provided for the purposes of localization or other similar needs. It is an optional file used to override defined behavior within the application. It is not needed for many implementations.

Overview of Localization in Agentry

Agentry applications provide full support for localization of the mobile application without directly modifying the definitions of the application project. This includes localizing the application for one or more languages with a single Server deployment, as well as localization for the purposes of industry- or deployment-specific terminology. This support is provided using override files referenced by the Server. An application is localized based on certain options in the `Agentry.ini` configuration file, and by making use of the localization override files.

Localization base files are files created by the Agentry Editor during a publish. Contained within them are the display values and enable switches that allow the localization of an application. All localization base files include the text `Base` in the name, as in: `ApplicationText`**`Base`**`.ini`. The publish wizard provides a check box allowing you to explicitly create these files.

Localization base files are copied and modified, creating the localization override files. The contents of these files are intended to override the defined values of the application. These files are read and processed by the Server upon startup. The override values are then sent to Clients during synchronization, after application definitions are synchronized but before production data is processed.

The different base files are intended for different aspects of the application and the Agentry software components. The initial contents of the base files are based on the definition of the application. The values contained in the files are those defined to display to the user. This

allows the translator or implementor to view the current values and to override them with the localized values for an implementation.

The localization files created for an application are reusable for other implementations of the same application. While customizations are normally a part of the implementation process, localization override files created for the standard deployment of the application provide a good starting point. If managed properly, override files can be created for an application in multiple languages. The files can be created a single time, and used repeatedly for deployments with the same general localization requirements.

Localization Configuration Options

There are several options within the `[Configuration]` section of the Agentry Server's properties that can impact the behavior of the localization functionality. These items can be configured using the SAP Control Center. The following options are those that pertain to the localization behavior:

- **localizations:** Use to specify which languages are supported by the application. This controls which override files the Server looks to read in when it is started. This option may or may not be needed, depending on how the application is being localized.
- **localizationsPath:** An optional setting used to specify where the language-specific override files are located. The default is a sub-folder to the Server's installation folder named `localizations` and is used when this setting is not listed or has no value set.
- **Override File Names:** There are several additional options in this section you can set to the name of the localization file for a specific area of the Client's UI. The impact of these settings on the Server's behavior differs depending on how localization is implemented. These settings are:
  - **applicationTextFile**
  - **clientStringsFile**
  - **applicationGlobalsFile**

The `[Configuration]` section contains additional options to those listed here, including reference to certain override files (`enableOverrideFile`, `transmitConfigurationFile`). Only those options discussed here pertain to localization behavior.

Localization Files

There are two types of files to work with when localizing an application: localization base files and localization override files. Both types of files are formatted in plain text. Open and edit these files in a standard text editor.

Localization base files can be created automatically as an option during publish of the application project from the Editor to the Server. The contents of these files are based on the definition of the application. These, then, are the base files from which override files are created.

Localization override files are those created by a translator using the localization base files as a starting point. The final contents of the override files are the localized or translated display values for the application. Each localization base file results in one or more override files.

Localization Base File List

The following files comprise the localization base files created by the Agentry Editor:

- **`ClientTextBase.ini`:** This base file contains display values that are a part of every Agentry Client regardless of the application deployed. Values are displayed in the menus and built in screens of the Client. The file contains a single section named `[Strings]`.
- **`GlobalsBase.ini`:** This base file contains the group, name, and defined values of all globals defined within the application. Each global group has a corresponding section denoted as `[`*`GlobalGroupName`*`]` within the file. Listed within a section are all the globals in the group and their defined values. Changing the value in this file will override the value of the global sent to the Clients.
- **`ApplicationTextBase.ini`:** This base file contains one section for each module defined within the application denoted as `[`*`ModuleName`*`]` and an additional section for the application definition itself, named `[Misc]`. Every definition within a module containing a display name, label, or caption attribute is listed in this file. The key portion identifies the specific definition and the value contains the display name text. Changing the value in the override file overrides the display name for that specific definition.
- **`TransmitConfigurationsBase.ini`:** This base file is not a part of localization as it relates to translation. Its usage is described in the information on Client-Server communications within Agentry.
- **`EnablesBase.ini`:** This base file contains one section for each module defined within the application, denoted as `[`*`ModuleName`*`]`. Each section lists all actions, pushes, detail screen fields, and list screen columns defined within the module. The value of each item is either `true` or `false`. It is likely that most of the values are set to `true`, unless the definition in question is defined to be disabled in the application. The override file created from this base file can contain items with a value of `false` to disable the corresponding definition or `true` to enable it. Disabling a definition will have a different impact depending on the definition type. Disabled fields and columns are not displayed on the Client. Disabled actions are always disabled and users cannot execute them. A disabled push will result in all behaviors related to that push being disabled. If there are no enabled pushes, and background sending is not enabled, users will not remain logged into the Server regardless of the transmit configuration definition.

Working With Localization Override Files

When creating the localization override files it is important to understand that the files are independent of one another. If all planned overridden values exist in one file, it is not necessary to create the other override files. Only those files containing values to be overridden are necessary.

Items within a single file are also independent values. A single override file only needs to contain the items for values to override. If other values within the file continue to use those settings defined in the application project, it is not necessary to list them in the override file.

In both cases, the defined display values and/or behaviors will always default back to the application definitions within the agentry application project. Omitting a particular override will not cause issue with the application behavior on the Client.

Localization Methods

When localizing an application, first determine the best method of localization. This determination is based on the needed language support, specifically whether multiple languages are needed for the same deployment or just a single language for all users.

If all users require the same localization overrides (i.e., a single language) use the *single language localization method*. If multiple languages are necessary for a deployment, you must use the *multi-language localization method*. You can also use the multi-language method in deployments where only a single language is needed. Do this to provide for future needs where multiple languages may be provided for the same deployment.

The primary differences between these two methods of localizing an application are:

- The Single Language method requires a single set of localization override files; the multi-language method requires one set of override files for each language.
- The multi-language method provides for the same base language with multiple locales. As an example, implementing Spanish as the base language, with two locales of Chile and Mexico.
- The single language method displays the same override values on all Clients, regardless of the devices' locale settings; the multi-language method uses the Clients' locale settings to determine the required override values.
- Use of the multi-language method requires knowing the proper language name and locale, collectively referred to as the *locale name*, for each of the languages supported in the deployment. This method also requires the configuration of client devices with the proper locale settings. The single language method does not make use of locale settings and therefore does not require this information.

Regardless of which method is used, the localization override files are created in the same manner. The localization method used dictates the proper names for the override files, as well as their proper location on the file system. The contents and format of the override files for a given language are the same for either method.

Configuring Back End Communications

The procedure for configuring the communications between the Agentry Server and the back end system is dependent on the specific system to which the Agentry Server will connect.

Configuring SQL Database System Connections

Prerequisites

The following items must be addressed before performing this procedure:

- Obtain and record the numeric identifier of the SQL Database system connection definition from the Agentry Editor.
- Obtain or create the user ID and password to be used to connect the Agentry Server to the database system.
- Obtain information on, or create and configure the necessary connection settings on the host system for the Agentry Server. This may be the System Data Source Name within ODBC for a SQL Server database, or the Local Net Service Name for an Oracle database.
- Determine whether or not users will be authenticated during synchronization via security information provided by the database.
- Perform a publish from the Agentry Editor to the Agentry Server. This will create the initial configuration sections for the SQL Database system connection that will then be modified further in this procedure.

Task

This procedure provides the steps for configuring the Agentry Server to connect with a database back end system using SQL for synchronization. This procedure provides a typical implementation example. Certain settings within the `[SQL-n]` section may be modified in relation to implementation-specific behaviors or needs. These items can include:

- User Authentication and Security.
- User auditing requirements and related settings.

These functional areas and optional behaviors are provided in the information on these topics. Changes to the system connection sections in the `Agentry.ini` file should always be made using the SAP Control Center.

1. This process begins by publishing the application project from the Agentry Editor to the Agentry Server. This will create the section `[SQL-n]` for the SQL Database system connection. It will also automatically add the proper option to the `[System Connections]` section.
2. Start the SAP Control Center. Connect to the system where the SAP Mobile Platform with the running Agentry Server is installed.
3. In the navigation pane of SAP Control Center, expand the Applications node and select the Agentry application.
4. In the administration pane, click the Configurations tab.

5. Select the section SQL-n, where n corresponds to the system connection definition in the Agentry application project. Click the Properties button above the list of configuration sections.

   This displays a list of the settings for the system connection to a database.
6. Edit the settings on this screen to allow the Agentry Server to connect to a database system.

   In this list the settings are set as shown in most cases, regardless of the implementation environment. The following lists those settings that are set specific to the environment, along with a description of their values:
   - `disconnectFromDB`, `disconnectTime`, and `reconnectTime`: These three settings are used when the Agentry Server should disconnect from the database once every 24 hours to allow for any batch processing. When `disconnectFromDB` is set to `true`, the Agentry Server will log out of the database server at the time specified by `disconnectTime`. It will log back in at `reconnectTime`. Users will not be able to synchronize their Clients during the period when the Agentry Server is logged out of the database.
   - `enableAuthentication`, `enablePreviousUserAuthentication`: Setting these options to `true` will enable the back end authentication functionality, requiring users to be authenticated against the database in order to perform synchronization. At least one system connection within a given application must perform back end authentication.
   - `dbConnectionName`, `dbConnectionUserID`, `dbConnectionPassword`: These three options configure the database to which the Agentry Server is to connect and the user ID and password to use. The `dbConnectionName` setting contains either the ODBC System DSN (MS SQL Server) or Local Net Service Name (Oracle) representing the database server.
   - `dbConnectionType`: This option is set to either `ODBC` or `Oracle`. `ODBC` is set when the connection between the Agentry Server and the database is made over ODBC, as is the case with MS SQL Server. `Oracle` is set for connections to an Oracle database server.
7. If multiple SQL Database system connections are defined for the application, repeat the preceding steps for each such connection type. When the system connections have been modified as needed, click the **[Apply]** button to apply the changes to the Server. Click **[OK]** to apply the changes and close the Edit Configuration Settings screen.

The Agentry Server will be configured to open and maintain a connection with the target database. The Agentry Server will make this connection when the changes are applied. The application being deployed will be able to synchronize data using ANSI SQL with the database for both upstream and downstream purposes. Based on configuration options, users may also be authenticated against this database system.

Next

Once this procedure has been completed successfully, the following additional tasks may need to be performed:

- Additional configuration tasks related to the application's synchronization with the target database may need to be performed. These can include modifications to the query initialization (`SqlBe.ini`) file and the query constants (`oracle_sd.ini` or `sqlserver_sd.ini`) files.

SqlBE.ini Query Initialization File

The configuration file `SqlBE.ini` for the Agentry Server contains several sections. This file is referred to generically as the "query initialization file." Each section corresponds to an event related to the Server's communications and synchronization with a SQL database. The settings for each of the sections consists of one or more SQL scripts to be processed by the Server when the event that corresponds to the section occurs.

The `SqlBe.ini` is the default query initialization file provided by the Server installer. It can be modified for an application, or it can be used as a template to create other query initialization files for applications with multiple SQL Database system connection types. Within the `[SQL-n]` section of the `Agentry.ini` configuration file, the option `queryInitFile` contains the name of the query initialization file to be used by the system connection.

File Format

The sections within the query initialization file list one or more SQL scripts to be run. These scripts are listed as key and value pairs, just as with any other configuration options in Agentry. However, the key portion of these options are unnamed keys. This means the Server is not looking for a specific configuration option name. Rather, all of the scripts within a section are processed in the order listed when that section's event occurs. The key may be any value desired. The common practice is to use key values of `q01`, `q02`, `q03`, etc. The numeric portion of these keys has no impact on the behavior of the Server, but is useful for those reviewing or modifying the query initialization file.

As an example, the `[LoggedIn]` section may appear as follows for an application:

```
[LoggedIn]
q01=InsertUserAudit.sql
q02=UpdateUserLastLoginTimestamp.sql
```

Application-Specific Changes

The query initialization files are normally configured initially by application developers or as the result of development efforts and standard application needs. Therefore, changes to these settings should only be made with a full understanding of how such changes will impact the overall behavior of the application. It is often the case that changes are made not to this file, but rather to the scripts it references.

Specific changes to either the settings in this file or to those scripts it references are made as the result of specific implementation requirements. Items such as user validation or auditing requirements can dictate when and what modifications may be necessary.

SQL Script Locations

The scripts referenced in the `SqlBe.ini` file must be located in one of two directories, both of which are sub-directories of the Agentry Server within the SMP installation location (e.g. `C:\SAP\MobilePlatform\Servers\AgentryServer\`).

The following two sub-directories are where the SQL scripts are located:

```
sql
sql\custom
```

The purpose behind having two separate folders for script files is to provide for standard and customized processing. For an application implementation, the `sql` folder will contain the standard scripts provided with the application installer. Changes to these scripts should be made to copies. These new versions should then be saved in the `sql\custom` folder.

The behavior of the Agentry Server when processing scripts listed in the `SqlBe.ini` file is to first look for the script file in the `sql\custom` folder. If found, the script will be processed and executed. If it is not found in the custom folder, the Agentry Server will then look for the same file in the `sql` folder.

This behavior allows for a script to be overridden while at the same time leaving the original version easily accessible. If it is desired to return to the default processing, the script file in the custom folder can simply be moved or renamed. The next time the Agentry Server executes the script, the default version will then be run.

Sections Listing

The following is a list of the sections within this file and the events to which they relate:

- `[Misc]` - This section should not be modified. This section contains two queries to be run by the Agentry Server, which are used for internal purposes.
- `[DbConnect]` - This section can contain a listing of SQL scripts to be run when the Agentry Server connects to the database. This connection is made when the Agentry Server is started, and also when the Agentry Server reconnects to the database based on the `disconnectFromDB` setting in the `[SQL-n]` section for the system connection.
- `[DbDisconnect]` - This section can contain a list of SQL scripts to be run when the Agentry Server disconnects from the database. This occurs as a part of the Server's shutdown processing, and also when the Agentry Server disconnects from the database based on the `disconnectFromDB` setting in the `[SQL-n]` section for the system connection.
- `[DbConnectionInit]` - SQL scripts listed in this section are run when the Agentry Server creates a connection to the database. This differs from the `DbConnect` section in that `DbConnectionInit` scripts are run whenever the Agentry Server creates a new

connection to the database, not just during startup and reconnect. New connections may be made whenever a user synchronizes the Agentry Client, or when the Agentry Server adds a connection to the shared connections pool.

- `[ValidateUser]` - This section can contain a list of SQL scripts to be run to validate a user against the database during client-server synchronization. These steps are the primary processing for user authentication with a SQL Database system connection. These scripts are run after those listed in `[DbConnectionInit]` and before any step definitions for the application are processed. If the scripts listed here return one of the failed validation values the user will be logged out of the Agentry Server. No application data will be synchronized in this situation.
- `[ValidatePreviousUser]` - This section can contain a list of SQL scripts to be run to validate a previous user against the database during client-server synchronization when a user change has occurred on the Agentry Client. These scripts are run after those listed in `[DbConnectionInit]` and before those listed in `[ValidateUser]`. If the scripts listed here return one of the failed validation values (discussed in the section on security) the previous user's data (pending transactions on the Agentry Client) will not be synchronized. In this situation, a user change cannot be completed until the previous user's data has been successfully synchronized.
- `[LoggedIn]` - This section can contain a list of SQL scripts to be run after a user has been logged in successfully; that is, the scripts listed in the `[ValidateUser]` section have successfully validated to the user.
- `[LoggedOut]` - This section can contain a list of SQL scripts to be run just prior to a user logging out of the server, when client-server synchronization has completed.
- `[UserInfo]` - This section can contain a list of SQL scripts to be run after those listed in the `[LoggedIn]` section have been run, and before the step definitions for the application are processed. These scripts are expected to return values that are then accessible to all other SQL scripts run against the system connection for a user. This includes SQL step definitions within the application itself. The values are available via the Syclo Data Markup Language using the data tag `<<user.info.`*`valueName`*`>>`.
- `[ApplicationGlobals]` - This section can contain a list of SQL scripts to be run to retrieve override values for global definitions within the application. These values are for the application in general and not considered to be user-specific.
- `[UserGlobals]` - This section can contain a list of SQL scripts to be run to retrieve override values for global definitions within the application. These values are for a specific user or user group and allow for overrides on a per-user basis.
- `[LoginFailed]` - This section can contain a list of SQL scripts to be run when a user fails validation, i.e., when scripts in the `[ValidateUser]` or `[ValidatePreviousUser]` indicate the user has failed validation. This section is processed when validation fails for any reason, including the user being blocked.
- `[LoginBlocked]` - This section can contain a list of SQL scripts to be run when a user fails validation because they have been blocked from having access. The scripts in `[ValidateUser]` or `[ValidatePreviousUser]` must explicitly indicate the

user has been blocked. In this case, these scripts are run after those in `[LoginFailed]`.

- `[ChangePassword]` - This section can contain a list of SQL scripts to run to process a password change by the user. These will be run after a user receives notification of an expired password or when a password is about to expire. These scripts should then perform the necessary processing to change the user's password. The scripts are expected to return a value indicating success or failure in processing the password change.
- `[ChangePasswordFailed]` - This section can contain a list of SQL scripts to run when a user password change fails. These scripts can return messaging to be displayed on the Client providing the user with information on the reason the new password was rejected.
- `[EnablePush]` - This section can contain a list of SQL scripts to run when a user performs a transmit and stays logged in for push processing. Specifically, these scripts are run when all of the following conditions are met:
  - The Agentry Client remains connected after synchronization as the transmit configuration definition is defined to enable push functionality.
  - A Push definition exists within the application.
- `[DisablePush]` - This section can contain a list of SQL scripts to run when a client previously connected to the Agentry Server to receive push data is about to disconnect. These scripts are run prior to those listed in the `[LoggedOut]` section.
- `[TimeZone]` - This section can contain a list of SQL scripts to run to determine the time zone of the database. The time zone returned by these scripts will be used when converting date and time values from one time zone to another, based on the database and Agentry Client time zones.

Query Constant Files for SQL System Connections

Query constant files are configuration files containing constant values that may be referenced by any SQL script run by the Agentry Server. Each SQL Database system connection can have its own query constant file. By default, there are two such files installed with the Server:

- oracle_sd.ini
- sqlserver_sd.ini

As the names imply each is intended for use by a system connection to one of the Oracle or SQL Server database systems. The values contained in these scripts are intended to be used in making SQL commands, either from definitions within the application or those run from the query initialization file (`SqlBe.ini`), more portable. Using the values listed within these files, a single SQL statement may be written that will execute properly in any of the database systems just mentioned.

The values within these files are accessible within a SQL statement run by the Server via SDML data tags. Following is a list of the values contained in the default versions of these files. Like all other configuration files for the Server, the query constant files can contain

multiple sections. By default they contain the single section `[Database]`, which contains all of the following options:

- `name` - The name of the database type. This value should not be changed.
- `getSystemTime` - This value is the database function called to return the current date and time.
- `timeStampFormat` - This is the format of a date and time value such that it can be inserted into a table column of a date and time data type. This is also the format used by the SDML processor within the Agentry Server when a date and time SDML data tag is expanded. Therefore, this should not be edited except in exceptional circumstances.
- `dateFormat` - This is the format of a date value such that it can be inserted into a table column of a date or date and time data type. This is also the format used by the SDML processor within the Agentry Server when a date data tag is expanded. Therefore, this should not be edited except in exceptional circumstances.
- `timeFormat` - This is the format of a time value such that it can be inserted into a table column of a time or date and time data type. This is also the format used by the SDML processor within the Agentry Server when a time data tag is expanded. Therefore, this should not be edited except in exceptional circumstances.
- `tempdate` - This is a temporary date value of 1/1/1901 12:00:00. This value is returned in the proper format for the database type to use it as a date and time value. This date and time value is assumed to be one that is not likely to be set for real data. If this default value is a valid date and time for the system it should be edited to one that is not valid.
- `substring` - This is the database function called to return a range of characters from within a larger string.
- `stringcat` - This is the database function or operator that will concatenate two string values.
- `charFunction` - This is the database function to convert a value to a character data type.
- `nullFunction` - The database function called to determine if a value is null.
- `singleRow` - Oracle databases require a select statement to include a from clause. When selecting literal values, e.g. SELECT 'X'..., the table object DUAL is used for this purpose in Oracle. No such object exists in SQL Server. The `singleRow` value in the Oracle constant file contains the text FROM DUAL. In the SQL Server script it contains an empty string.
- `unicodePrefix` - This is the value prefixed to a unicode value, identifying it as such to the database.
- `terminalErrorCodes` - This is a semi-colon delimited list of return values treated as terminal error codes by the Agentry Server. When such an error is returned, the Server will end the synchronization with the Client, returning an error message.
- `retryWithChangeErrorCodes` - This is a semi-colon delimited list of return values that may be received from the database system as the result of processing a SQL statement. The values listed here will be treated as a retry with change error for the purposes of

transaction failure handling. If this functionality is not enabled in an implementation, this setting has no affect on the Server's behavior.

- `retryWithoutChangeErrorCodes` - This is a semi-colon delimited list of return values that may be received from the database system as the result of processing a SQL statement. The values listed here will be treated as a retry without change error for the purposes of transaction failure handling. If this functionality is not enabled in an implementation, this setting has no affect on the Server's behavior.
- `fatalWithMessageErrorCodes` - This is a semi-colon delimited list of return values that may be received from the database system as the result of processing a SQL statement. The values listed here will be treated as a fatal with message error for the purposes of transaction failure handling. If this functionality is not enabled in an implementation, this setting has no affect on the Server's behavior.
- `fatalWithoutMessageErrorCodes` - This is a semi-colon delimited list of return values that may be received from the database system as the result of processing a SQL statement. The values listed here will be treated as a fatal without message error for the purposes of transaction failure handling. If this functionality is not enabled in an implementation, this setting has no affect on the Server's behavior.

Additional Query Constant File Sections

In addition to the standard `[Database]` section of the query constants files, other sections may be added for application-specific purposes, such as:

- The application extends a back end system that may be deployed on different database systems, and certain values or functionality is different within those systems.
- The application extends a back end system deployed in multiple languages. As a result, certain constant values, such as Y and N flags, are different in different languages. In this case, the values can be represented as options here, and these would then be the items referenced via SDML in the SQL statements. When deployed in a different language, the constant file can be updated with the proper language-specific value.
- Additional functions may be needed beyond those in the standard version of the query constants file. In this case it may be considered cleaner to add a separate section to differentiate between those functions added after the fact, and those provided by default.

In almost all situations the contents of this file are modified by, or at the direction of the application developer, as the values are referenced as a part of the application's business logic. Changes made at implementation time should only be performed with an understanding of the affect such changes will have and the ways in which the configuration options are used. Furthermore, the default values provided with the Agentry Server should only be changed in the rarest of situations, particularly those related to the formats of date and time values.

Configuring the Query Constant Files for a System Connection

It is possible for a single SQL Database system connection to make use of multiple query constant files. It is also possible to change the name of the file referenced for the connection. This is accomplished by changing the configuration option `queryConstantFiles` in the `[SQL-n]` section within the `Agentry.ini` file.

If a different file name is to be used, or if multiple constant files are needed for a system connection, the `queryConstantFiles` configuration option should be modified to reference this. For multiple file listings, each file name is separated by a semi-colon.

```
-- different file name
[SQL-1]
...
queryConstantFiles=MyConstantFile_sd.ini
...
--- multiple constant files
[SQL-1]
...
queryConstantFiles=oracle_sd.ini;MyApplicationConstantFile_sd.ini
```

If a different file is to be used from the default file, that file must contain the `[Database]` section, including all configuration options.

Configuring Java Virtual Machine System Connections

Prerequisites

Before executing this procedure the following items must be addressed:

- Obtain and record the numeric identifier for the Java Virtual Machine system connection definition from the Agentry Editor.
- Obtain the following information on the back end system with which the Agentry Server is to connect:
  - Host name and port numbers for the system hosting the Java application server.
  - Java application server name or identifier.

  Coordination with network administrators, as well as those with information regarding the back end system prior to performing configuration tasks is also recommended.
- Note the full path of the installation folder for the Agentry Server. In the following instructions the path `C:\SAP\MobilePlatform\Servers\UnwiredServer\Repository\Agentry\default\<AppName>` will be used.
- Obtain any `.jar` or `.class` files containing business objects with which the application being implemented may use. These are normally provided with the back end system with which the Agentry Server is synchronizing. These files must be made accessible to the Agentry Server.
- Location of and access to the JDK and JRE (installed previously).
- Determine whether or not users will be authenticated using the JVM system connection. In such a situation, users will not be able to synchronize unless they are successfully authenticated. The proper user credentials must be established within the back end system prior to allowing users access to the system. This may be done before or after the following procedure.

Task

This procedure provides the steps for configuring the Agentry Server to connect with a Java application server back end system using Java logic for synchronization. This procedure provides a typical implementation example. Certain settings within the `[Java-n]` section may be modified in relation to implementation-specific behaviors or needs.

1. This process begins by publishing the application project from the Agentry Editor to the Agentry Server. This will create the section `[Java-n]` for the Java Virtual Machine system connection. It will also automatically add the proper option to the `[System Connections]` section.
2. Start the SAP Control Center. Connect to the system where the SAP Mobile Platform with the running Agentry Server is installed.
3. In the navigation pane of SAP Control Center, expand the Applications node and select the Agentry application.
4. In the administration pane, click the Configurations tab.
5. Select the section Java-*n*, where *n* corresponds to the system connection definition in the Agentry application project. Click the Properties button.

   This displays a list of the settings for the system connection to a database.
6. Edit the settings on this screen to allow the Agentry Server to connect to a system via a Java interface.

   In this section many of the settings are set as shown in most cases, regardless of the implementation environment. The following lists those settings that are set specific to the environment, along with a description of their values:
   - `classPath`: This setting contains several paths used by the Agentry Server in relation to processing Java logic for the application. The path information here is appended to the system variable CLASSPATH. Any values in either location are accessible to the JVM of the Agentry Server. The values for the `classPath` setting should represent the following information:
     - The location of the Java Development Kit `lib\tools.jar` file.
     - The location and file name of any `.jar` or `.class` files needed by the application's Java logic.
     - The same path value as listed in the setting `outputDirectory`.
     - The full path of the installation folder for the Agentry Server, appended with the text `Java`, as in `C:\SAP\MobilePlatform\Servers\UnwiredServer\Repository\Agentry\default\<AppName>`.
   - `outputDirectory`: This is a path value that controls where the compiled output is stored for execution. These are the `.class` files resulting from compilation of the `Steplet`, `ComplexTable`, and `DataTable` classes defined within the application. This is normally the installation location of the Agentry Server, as in `C:`

\SAP\MobilePlatform\Servers\UnwiredServer\Repository\Agentry\default\<AppName>.

- `sourceDirectory`: This is a path value that specifies where the `.java` files for the application are to be placed when extracted from the definitions. These files are then compiled at run time by the JVM and the resulting `.class` files are stored in the `outputDirectory` location. This is normally the installation location of the Agentry Server, as in `C:\SAP\MobilePlatform\Servers\UnwiredServer\Repository\Agentry\default\<AppName>`.
- `deleteSource`: This option is set to either true or false. When true, the `.java` files stored in the `sourceDirecotry` will be deleted after they have been compiled into the `.class` files stored in the `outputDirectory`. This is normally only set to true for a development environment, where the Java code is being modified regularly as a part of the development cycle.
- `enableAuthentication`: Setting this option to `true` will enable the back end authentication functionality, requiring users to be authenticated via means defined for the JVM system connection in order to perform synchronization. If multiple system connections are in use for an application, at least one must perform the back end authentication.
- Other options within this section include settings for the JVM itself, allowing for optimization of the environment for the specific application. These are normally set by developers, or at their direction. Specifics of these are covered in the sections on JVM system connection optimization.

7. If multiple JVM system connections are defined for the application, repeat the preceding steps for each such connection type.

Once this procedure is complete the Agentry Server will be configured to connect with a JVM System Connection. The Server will be able to perform synchronization tasks for both upstream and downstream processes through the target Java application server. Depending on configuration, users may also be authenticated through this system connection.

Next

Once this procedure has been completed the following tasks may need to be performed:

- A connectivity test can be performed by starting the Agentry Server and then performing a transmit from an installed Agentry Client or the Agentry Test Environment. The Server does not open a connection to a Java application server upon startup, but rather waits until a Client attempts to synchronize.
- Additional system connections of other types may need to be configured based on the needs of the application. This should be done before attempting to connect a client.

JVM Optimizations and Settings

Additional options within the `[Java-n]` section of the `Agentry.ini` file are provided to allow the implementors, administrators, and developers of an application control over the

behavior of the JVM run by the Agentry Server. These settings can be modified using the SAP Control Center. Before modifying these items be sure to review both the following descriptions, and also any information provided with the Java components installed to the Agentry Server's host system.

Following is a list of these settings and descriptions of their purpose and areas of behavior affected:

- `printStackTrace` - This option may be set to true or false. By default, any exceptions thrown by the Java logic are logged to the `events.log` file for the Agentry Server. Setting this option to `true` will also include the stack trace for that exception in this same log file.
- `printBusinessLogicStackTrace` - This option may be set to `true` or `false`. A special exception class, `JavaBusinessLogicError`, is provided with the Agentry Java API. Developers may write logic to throw this exception in certain circumstances. The `printBusinessLogicStackTrace` allows for the stack trace of this exception to be displayed on the transmit dialog of the Agentry Client. This functionality is provided primarily for development testing and debugging, and should only be set to `true` for these purposes.
- `initialHeapSize` -. This configuration setting provides the same behavior as the `-Xms` JVM parameter. If this parameter is specified in the `nonStandardJavaOptions`, the `initialHeapSize` setting should be omitted from the `[Java-n]` section. This option is set to a numeric value and represents the initial amount of memory in megabytes to be allocated by the JVM for the initial heap. If not specified the default value is 2MB. This should be set based entirely on the needs of the application. Also, restrictions related to the amount of memory that may be specified may differ from one version of Java to the next, and these restrictions should be reviewed prior to setting this option.
- `maxHeapSize` - This configuration setting provides the same behavior as the `-Xmx` JVM parameter. If this parameter is specified in the `nonStandardJavaOptions`, the `maxHeapSize` setting should be omitted from the `[Java-n]` section. This option is set to a numeric value and represents the maximum amount of memory to be allocated by the JVM, in megabytes, for the heap. The default value is 64MB if not specified. This should be set based entirely on the needs of the application. If heap memory exceeds this limit, an error will be encountered on the Agentry Server. Also, restrictions related to the amount of memory that may be specified may differ from one version of Java to the next, and these restrictions should be reviewed prior to setting this option.
- `reduceOSSignalUse` - This option may be set to `true` or `false`. When `true`, and only when the Agentry Server is running as a Windows Service, the JVM will remain running after the Agentry Server service is stopped. When the service is restarted, it will not start a new instance of the JVM, but use the one currently running. `True` is the recommended setting for this option whenever the Server is run as a Windows Service.
- `nonStandardJavaOptions` - This configuration option contains text representing the JVM parameters (command line options) sent to the JVM when started by the Agentry

Server. The options here may be any supported by the version of the JVM in use. Examples may include options related to garbage collection or debugging options.

- `lowMemoryPercentage` - This option specifies a percentage of the maximum heap size. When the available heap memory allocated by the JVM falls below this percentage, a warning message is printed to the `events.log` for the Agentry Server. The default value for this option is 5 percent of memory.
- `performCompile` - This option may be set to `true` or `false` and specifies whether the `Steplet`, `ComplexTable`, and `DataTable` Java classes of an application should be compiled every time they are loaded at run time. This option is likely to be set to `false` in all production environments, as changes to these classes occur infrequently. It may be `true` or `false` in the development environment, depending on the nature of changes performed. Much of the Java logic implemented for an application is not contained within the `Steplet`, `ComplexTable`, or `DataTable` classes and changes made may not directly impact these class implementations within the application. Therefore, compiling these classes during every transmit is not always necessary. In such situations this option may be set to false to prevent this unnecessary processing and improve efficiency.

Configuring HTTP-XML System Connections

Prerequisites

Before performing this procedure, the following items must be addressed:

- Obtain and record the numeric identifier for the HTTP-XML system connection definition from the Agentry Editor.
- The base URL of the HTTP-XML server to which the Agentry Server will make CGI or HTTP requests. This value will be prefixed to the paths of specific requests within the application, as defined in the Editor.

Task

This procedure provides the steps for configuring the Agentry Server to connect with a Web service back end system using HTTP-XML for synchronization. This procedure provides a typical implementation example. Certain settings within the `[HTTPXML-n]` section may be modified in relation to implementation-specific behaviors or needs. These items can include:

- User Authentication and Security.
- User auditing requirements and related settings.

These functional areas and optional behaviors are provided in the information on these topics. Changes to the system connection sections in the `Agentry.ini` file should always be made using the SAP Control Center.

1. This process begins by publishing the application project from the Agentry Editor to the Agentry Server. This will create the section `[HTTPXML-n]` for the HTTP-XML system

connection. It will also automatically add the proper option to the `[System Connections]` section.

2. Start the SAP Control Center. Connect to the system where the SAP Mobile Platform with the running Agentry Server is installed.
3. In the navigation pane of SAP Control Center, expand the Applications node and select the Agentry application.
4. In the administration pane, click the Configurations tab.
5. Select the section HTTPXML-*n*, where *n* corresponds to the system connection definition in the Agentry application project. Click the Properties button above the list of configuration sections.

   This displays a list of the settings for the system connection to a database.
6. Edit the settings on this screen to allow the Agentry Server to connect to a Web service.

   In the above example many of the settings are set based on the needs of the application and will not be changed at implementation. The following lists those settings that are set specific to the implementation environment, along with a description of their values:
   - `baseURL`: This is the base URL of the server to which the Agentry Server will make CGI or HTTP requests. This value may be the IP address or URL value. This value may be immediately followed by the port number, with the URL and port separated by a colon.
   - `constantsFile`: This is the name of the file containing constant values that may be referenced via SDML. The initial contents of this file are normally set as a part of the application development process, and are then modified for implementation needs. By default this file is expected to be located in the same folder as the Agentry Server.
   - `xmlNamespaces`: Listed in this option can be one or more of the namespaces involved in the data synchronization process. Each namespace listed here is separated by the pipe (|) symbol.
   - `xmlValidateOnParse`: This configuration option is set to true or false. It controls whether or not the Agentry Server and its XML parsing routines should validate the returned XML data against the XML schema or DTD. When true, this validation will occur and any violations of this schema will result in an error. If set to false, the schema is not validated and any violations will not impact the processing of the data returned.
   - `enableAuthentication`, `enablePreviousUserAuthentication`: Setting these options to `true` will enable the back end authentication functionality, requiring users to be authenticated against the back end system in order to perform synchronization. If multiple system connections are in use for an application, at least one must perform the back end authentication.
   - `httpConnectTimeout`: This value is set in seconds and specifies the amount of time the Agentry Server will spend while attempting to establish a connection to the HTTP server.
   - `authenticationCertificateStore; -Password; -PasswordEncoded`: These three options are related to the authentication

performed via authentication certificates with the back end. The `authenticationCertificateStore` lists the location and file name for a certificate store. The `authenticateCertificateStorePassword` contains the password to access that file. `authenticateCertificateStorePasswordEncoded` is the flag to indicate if that password is or is not encoded in the `Agentry.ini` file. This password is encoded using the Quick Password Encoder utility provided with the Server installation.

7. If multiple HTTP-XML Server system connections are defined for the application, repeat the preceding steps for each such connection type.

When this configuration is complete, the Agentry Server will be configured to connect to a Web Service server, with the ability to make cgi requests and process returned XML data. Data synchronization can be performed in this manner for both upstream and downstream processing.

Next

The following items may be necessary after completing this procedure:

- A basic connectivity test can be performed by starting the Agentry Server and then performing a transmit from an installed Agentry Client or the Agentry Test Environment. The Server will not open a connection at startup, instead making this connection when a Client connects to synchronize data via a transmit.
- If additional system connections are required by the application being deployed, those should be configured prior to starting the Agentry Server.

Configuring File System Connections

Prerequisites

Before executing this procedure, the following items and information should be addressed:

- Obtain and record the numeric identifier for the File System connection definition from the Agentry Editor.
- Determine whether or not users will be authenticated during synchronization by logging them into the host system.
- Obtain and record the Windows User Domain for user authentication.
- Determine the desired location for the Agentry Server to store temporary script files for execution. The default location is the Windows TEMP folder as configured for the system.
- Determine whether or not the previous user should be connected to the Windows system when a user change occurs on the Clients. It may be desirable to omit this portion of synchronization for the sake of expediency when file transfer functionality is in use.

Task

This procedure provides the steps for configuring the Agentry Server to connect to the host system. This procedure provides a typical implementation example. Certain settings within the `[File-n]` section may be modified in relation to implementation-specific behaviors or needs.

1. This process begins by publishing the application project from the Agentry Editor to the Agentry Server. This will create the section `[File-n]` for the File system connection. It will also automatically add the proper option to the `[System Connections]` section.
2. Start the SAP Control Center. Connect to the system where the SAP Mobile Platform with the running Agentry Server is installed.
3. In the navigation pane of SAP Control Center, expand the Applications node and select the Agentry application.
4. In the administration pane, click the Configurations tab.
5. Select the section File-*n*, where *n* corresponds to the system connection definition in the Agentry application project. Click the Properties button above the list of configuration sections.

   This displays a list of the settings for the system connection to a database.
6. Edit the settings on this screen to allow the Agentry Server to connect to host's file system.

   In this list many of the settings are set as shown in most cases, regardless of the implementation environment. The following lists those settings that are set specific to the environment, along with a description of their values:
   - `enableAuthentication`: This setting controls whether or not users are authenticated with the Windows system upon which the Agentry Server is running. When set to `true`, the user ID and password from the Client will be used to attempt to authenticate the user, meaning all users must have an account on the Server's host system.
   - `userDomain`: This setting contains the name of the domain to which users belong. If the `enableAuthentication` option is set to `true`, the value in `userDomain` determines which domain users are authenticated with. If `userDomain` is left set to an empty string, or is not present, the Agentry Server will attempt to determine which domain to use. It is recommended that this option be set to the proper domain, rather than being left blank.
   - `tempPath`: This configuration setting is optional and may be set to any valid path on the host system of the Agentry Server. When set to a valid path, any temporary batch files generated by the Agentry Server to be executed as a part of Client-Server synchronization will be stored in and executed from this location. If this setting is left blank or omitted the TEMP folder of the host system will be used.
   - `allowPreviousUserLogin`: This setting controls whether the previous user is logged into this system connection when a user change occurs on the Client. When this

is false, the previous user will not be logged into the file system, and any Step definitions for this system connection within the application are not executed.

7. Unlike other system connection types, there can be only one File System connection type defined for an application.

When this procedure is complete the Agentry Server will be configured to connect to its host system. This connection allows the Server to execute commands, read in file data, and transfer files as defined by an application. This can be down for both upstream and downstream synchronization.

Next

Once this procedure is complete the following items may be necessary:

- If any additional system connections of a different type are required by the application they should be configured before starting the Agentry Server.
- A basic connectivity test can be performed by starting the Agentry Server and verifying it is able to connect to the host operating system.

Configuring Client-Server Communications

Configure client-server communications based on the connection type: Agentry Clients using ANGEL protocol or Web Clients.

Configuring Agentry Client-Server Communications

Prerequisites

Ensure the following requirements needed for configuration of the ANGEL secure communications are met prior to modifying the configuration files for the Agentry Server:

- SSL authentication, including whether or not a different authentication certificate is needed for the Server. By default, a certificate is provided named `AgentryServer.pfx`.
- Determine if the Client requires authentication through SSL. If so, trusted root certificates are needed on the Server with matching entries for the authentication certificates installed on the Clients.
- Determine if the default time-out of 300 seconds and keep-alive duration of 60 seconds are adequate. If not, determine the proper values for these items as they are configured in this procedure.
- Retrieve and record the proper domain/IP address(es) and port number(s) from which the Agentry Server receives requests from Clients.

Task

This procedure describes the steps necessary to configure the Client-Server communications for the mobile application. Configuration of the ANGEL communications section is required for any deployment of an application. Many of the necessary settings for this connection type are implementation-specific. This process involves the modification of the `[ANGEL Front End]` and `[ANGEL Front End Ports]` sections of the `Agentry.ini` file. Always make changes to these sections using the SAP Control Center.

1. Start the SAP Control Center. Connect to the system where the SAP Mobile Platform with the running Agentry Server is installed.
2. In the navigation pane of SAP Control Center, expand the Applications node and select the Agentry application.
3. In the administration pane, click the Configurations tab.
4. Select the check box for **ANGEL Front End** and click the Properties button. Edit the settings on this screen to allow the Agentry Server to support the client-server communications for the implementation environment.

   Following are the settings that are configurable for these options:

   - **`trustedCertificateStore`:** Specifies the trusted certificate store containing the trusted certificate(s) used when client authentication is enabled (`authenticateClient=true`). This can be specified as a Certificate File (`.CER`) or Certificate Store File (`.SST`).
   - **`authenticationCertificateStore`:** Specifies the location of the Server's authentication certificate. This can be a Certificate File (`.CET`), Certificate Store File (`.SST`), or a Personal Information Exchange File (`.PFX`). The certificate identified here must be a trusted root certificate for the Agentry Clients.
   - **`authenticationCertificateStorePassword`, `authenticationCertificateStorePasswordEncoded`:** Password to access the authentication certificate identified in `authenticationCertificateStore`. Password encoded indicates whether or not the password listed here is encoded. This password is only encoded if `authenticationCertificateStore` is set to a value other than the default `AgentryServer.pfx`.
   - **`authenticateClient`:** Specifies whether or not the Agentry Client must provide an authentication certificate. This certificate must be traceable to a trusted root certificate, though intermediary authorities can exist.
   - **`timeout`:** Duration of time, in seconds, that the Agentry Server keeps a socket open between the Server and the Agentry Client without any activity. Once this limit is reached, the socket is closed.
   - **`keepAliveTime`:** Duration of time between keep-alive messages sent from the Server to the Client, preventing the time-out value from closing the socket. This

`keepAliveTime` is used only when background sending or push functionality is enabled for the application.

- **`minimum-`, `maximumCipherStrength`:** These two settings specify, in bits, the cipher strength of the data encryption used by this connection type. Leaving these items commented out (as shown above) or omitting them results in Windows determining the cipher strength.

5. Click **[OK]** to close the screen.

   The changes are saved and if necessary, the Agentry Server is restarted.

6. Open the `agentry.ini` file for the Agentry Server and search for the section `[ANGEL Front End Ports]`. You must initially manually edit this section. You cannot add new port options to this file through the SAP Control Center, though you can modify the settings using the SAP Control Center once they are added to the file. You can configure the Server to listen on one or more ports and network adapters. If multiple Agentry Servers are deployed for the same application, separate configurations are needed for each Server instance. These settings cannot be configured using the SAP Control Center for Agentry Servers within a cluster unless all Servers have the same port settings, which is typically not the case.

   ```
   [ANGEL Front End Ports] port1=7003 port2=127.0.0.1:7013
   port3=localhost:7080 port4=MyHostSystem:7020
   ```

   These ports must be free and can be specified by their port name. Whichever port is listed first in this section is used as the default port. All entries must include a port number and may include the host name or IP address. Finally, any IP addresses or host names listed here must have corresponding network adapters configured on the host system.

7. Review the modifications made to this file. When satisfied of their accuracy, save and close the `Agentry.ini` file.
8. Restart the Agentry Server in order for the modifications to take effect.

When this is complete, the communications between the Agentry Clients and Agentry Server are configured.

Next

After changing any communications settings, you must test the communications between the Client and Server. If multiple communications methods are employed, i.e., if there are multiple ports configured in the `[ANGEL Front End Ports]` section, test the connections from the Agentry Clients using each of the possible network addresses and/or port numbers.

Configuring Web Client Communications

The following procedure explains the configuration of the Agentry Server to act as a web server, allowing web browser client access to the application. This is done for BlackBerry Handheld Browser and Internet Explorer support. It involves the modification of the `[Web`

`Server Front End]` and `[Web Server MIME Types]` sections of the `Agentry.ini` file.

Prerequisites

The following items and tasks should be addressed prior to performing the procedure provided here:

- Determine the proper port on the host system for the Agentry Server to listen for web client requests. The default is 80 for HTTP and 443 for HTTPS (SSL) communications.
- Determine if SSL is to be used. If so, obtain the needed authentication certificates for the Agentry Server. Trusted root certificates of the client devices must match this authentication certificate. The default certificate contained in `AgentryServer.pfx` may also be used.
- Determine if any additional MIME types not configured on the client devices, and not contained in the default `Agentry.ini` file `[Web Server MIME Types]` section are needed. Review this section of the `Agentry.ini` file to obtain the default settings.

Task

1. Start the SAP Control Center. Connect to the system where the SAP Mobile Platform with the running Agentry Server is installed.
2. In the navigation pane of SAP Control Center, expand the Applications node and select the Agentry application.
3. In the administration pane, click the Configurations tab.
4. Select the check box for Web Server Front End. Here, the communications between browser clients and the Agentry Server are configured.

   The items modified in this section for most implementations are described next:
   - `listenOn`: This is the port number used by the Agentry Server to listen for requests made by browser clients. The default of 80 is used for non-SSL connections. If SSL is to be used, the default is 443. In either case the port number may be changed for firewall support or if the default ports are in use by other applications or processes.
5. If the MIME types must be modified, you must navigate to the installation location of the Agentry Server. Here open the `Agentry.ini` configuration file and search for the section `[Web Server MIME Types]`. These MIME types are for the Agentry Server and are not applied to the client devices. The Agentry Server makes use of those MIME types configured on the host system as well those listed in this section. If a type is defined in both locations, the configuration found in the `Agentry.ini` file will override the one configured on the host.
6. Once changes have been completed for the `[Web Server MIME Types]` section, you can close and save the `Agentry.ini` file.

Localizing Agentry Applications

Configure localization for Agentry applications based on the localization method (single or multiple languages) and any overrides required for specific terminology.

Localization: Single Language Support

Using the single language method, localization base files are generated through a publish of the application project. These files are then provided to a translator. The translator replaces the default display values found in these files with the translated text, creating the localization override files.

The localization override files are then placed in the installation folder of the Agentry Server. The localization override files should be named to match the corresponding `[Configuration]` section setting for the override file. Following are the default names of these files:

```
applicationStringsFile=ApplicationText.ini
```

```
clientStringsFile=ClientText.ini
```

```
applicationGlobalsFile=ApplicationGlobals.ini
```

If it is necessary to change the names of the localization override files, then you must modify the above-listed settings accordingly.

Once the localization files are created and loaded by the Server, Clients will receive any override values within these files. Any values not overridden come from the application definitions.

Use this method of localizing an application only when a single localized language is provided to all users, or when other localizations not related to an actual language translation are needed. The latter might include changing UI labels and other display strings based on deployment or industry specific terminology.

Single Language Localization Method

Prerequisites

Address the following items prior to following this procedure:

- The override files `ClientText.ini`, `ApplicationText.ini`, and `Globals.ini` must exist with the translated or overridden values. These files must be based on the localization base files generated for the current version of the application.

- Verify the values for the configuration options `ApplicationStringsFile`, `ApplicationGlobalsFile`, and `ClientStringsFile` in the section `[Configuration]` of the Server's `Agentry.ini` file. If a disparity exists between these settings and the file names, change either the settings or the file names so that both match. Accomplish this by either viewing and modifying the `Agentry.ini` file directly, or by using the SAP Control Center. If changes are made to the `Agentry.ini` file directly, make a backup copy prior to modifying the file.

Task

This procedure describes how to localize an application using the single language localization method. Use this procedure when translating an application's Client UI into a different language from the one in which it was developed. You can combine this procedure with the multi-language method if needed. However, this process is not discussed here, with the focus on the single language method of localization. This procedure is applicable for both language translation, as well as other localization requirements related to terminology or user interface overrides.

1. Copy the translated or localized versions of the localization override files to the folder for the Agentry Server instance for your application, i.e. `C:\SAP\MobilePlatform\Servers\UnwiredServer\Repository\Agentry\default\<AppName>`.
2. Restart the Server so that the localization files are read in.
3. Clients must perform a transmit with the Server to receive the user interface definitions with the override values displayed.

Once this procedure is complete, the Clients will display the overridden, localized values on all related user interface components.

Localization: Multi-Language Support

When localizing an application for multiple groups of users, each with different language or localization requirements, you must use the multi-language method of localization. This localization method allows for different localization override values to be used for different groups of users without the need for multiple Agentry Server installations.

With multi-language support, multiple sets of localization override files are used, with each set containing the translations for one of the languages. The localization files are then made available to the Server. During transmit, Clients receive the override values from one of the sets of localization files based on the client device's locale settings.

You must name the localization files based on the settings in the `[Configuration]` section of the `Agentry.ini` configuration file for the Server. There are two additional settings within this same section related to the multi-language localization method. All localization settings are listed here, followed by a discussion of how the localization override files are named.

- localizations
- localizationsPath
- applicationStringsFile
- applicationGlobalsFile
- clientStringsFile

The localizations setting contains one or more *locale name* values. A locale name is defined as a text value containing the language name and local of a supported language for the application. The language name indicates the language used (e.g., English, French, Spanish). The locale is the sub-language, dialect, or region for the language (e.g, United Kingdom, Canada, Mexico). The combination of the language name and locale produces the *locale name* used by the Server to identify the language overrides for a particular Client.

The supported languages must be listed in the localizations setting of the [Configuration] section of the Agentry.ini file. Each language is identified by its locale name. The values for this setting are separated by semi-colons when multiple language names are listed. The specific values are the locale names for the desired languages as listed on the Microsoft Developer Network web site at the following URL:

```
http://msdn.microsoft.com/en-us/library/ms776260(VS.85).aspx
```

This page contains a table with several columns related to language names, locales, and similar information. The column whose value is important here is the one labeled **Locale Name**. Each locale name consists of the language name followed by the locale, with the two values separated by a hyphen. For example, Spanish is represented by the language name es. There are several different locales for Spanish. Two of these are Spain and Mexico, which are represented by the locale values ES and MX, respectively. The resulting locale names are es-ES for the Spain dialect of Spanish and es-MX for the Mexican dialect. To support these two locales within the application, list the two locale names in the localizations configuration option.

The actual override, or translated values, are contained in the localization override files. When using the multi-language localization method, one set of files is created for each locale name listed in the localizations setting. The names of these files must include the locale name to which they pertain. Specifically, the locale name must come after the other portions of the file name and before the extension. The locale name is separated by periods. Following is an example of the ApplicationText.ini file name for the Mexican-Spanish locale name:

```
ApplicationText.es-MX.ini
```

The es-MX portion of the name indicates that the overrides in this file are provided for support of the Spanish language, Mexican dialect locale. This locale name must be the same as the one listed in the localizations setting. The other localization file names are of the same format.

Another impact on the file name is the setting in the [Configuration] section of the Agentry.ini file. If the name of the override files are changed in this setting, that alters which file name the Server looks to read in when started. Using the

`ApplicationText.ini` file as an example, this file is named in the setting `ApplicationStringsFile`. If the value of this setting is changed to `AppTxt.ini`, the localization override file name must reflect this value. In this case the file name would be:

```
AppTxt.es-MX.ini
```

While it is possible to change the settings of the `Agentry.ini` file to look for different file names, this is not a recommended practice in most situations. The ability to change these names is provided for the sake of comprehensiveness. Unless dictated by specific implementation or deployment needs, it is recommended that the localization override files are named to match the default settings for these options.

An additional option for supported languages is to provide for a base language that is used without concern over a locale. In this case, the language name of the language is listed in the `localizations` option. An example of a base language is simply Spanish, which has a language name of `es`. Specify a base language for one of two reasons:

1. It is possible that a Client will send a locale name that is not listed in the localizations option.
2. It may not be necessary to distinguish between one local and another. Simply supporting a language is sufficient.

If a base language is listed in the localizations option, Clients whose language name portion of the provided locale name match that base language will receive the corresponding overrides. As an example, assume the following option is set as shown:

```
localizations=es-ES;es-MX;es
```

The supported languages, then, are: Spanish as spoken in Spain; Spanish as spoken in Mexico; and Spanish. If a Client's locale name is transmitted to the Server as `es-ES` or `es-MX`, that Client will receive the corresponding overrides from the localization files for the specified dialect. If, however, a Client's locale name is `es-PA`, which is the locale for Spanish - Panama, neither of the first two configured locales match. If no other option is provided, that Client will receive the defined display values as set in the application project.

However, since the language name `es` is listed in the `localizations` setting, the Client will receive the override values from the localization files for Spanish. This, results in the Client receiving overrides that, while they are not ideal for the users' native dialect, are likely to be far better than the defined language of the application, which may not be in Spanish but in English.

The format of the localization file names for such a base language is similar to those discussed previously. The difference being that any locale portion is omitted from the name.

```
ApplicationText.es.ini
```

The above example contains the language name `es`. The overrides in this file are the ones sent to Clients providing locale names for Spanish, but not Spain or Mexico.

It is also possible to use both the single language and multi-language methods for the same implementation. In this case, the override files provided for the single language method, that is, the `ApplicationText`, `ClientText` and `ApplicationGlobals` files that do not include locale names, act as default values.

Assuming that the definitions are created in one language, with overrides provided using both the single language and multi-language methods, the following list provides the order used to determine what value is sent to the Client for display. This order of precedence also applies to values overridden in one localization file but not another, or values not overridden at all.

1. The Server first attempts to exactly match the locale name received from the Client to the options in the `localizations` setting. If a match is found, the corresponding localization files are used and all values in these files are sent to the Client. If a match is not found for the Client's locale name in the `localizations` configuration option, or if one or more values are not overridden in these files, then...
2. The Server determines if the base language of the Client's locale name is listed in the `localizations` setting. If a match is found for the language name, the corresponding override files are used. Any values found in the base language files are not used if they are already overridden by the more specific locale name override files. If a match is not found for the Client's base language in the `localizations` configuration option, or if one or more specific display values are not overridden by these localization files, then...
3. The single language override files and any override values in the files, if present, are processed. Any overlaps for individual override values between these files and those localization files processed before them are not used. If the single language override files are not present, or if one or more specific display values are not overridden by these localization files, then...
4. The defined values from the application project are used.

Multi-Language Localization Method

This procedure describes how to localize an application using the multi-language method. Use this procedure when translating an application's Client UI into one or more different languages from the one in which it was developed. You can combine this procedure with the single language method if needed. However, that process is not discussed here, with the focus on the multi-language method of localization.

Prerequisites

Address the following items prior to performing this procedure:

- Determine the supported languages for the application and obtain the language names and locale names to match. For a complete list of locale names, see the URL:

  ```
  http://msdn2.microsoft.com/en-us/library/ms776260(VS.85).aspx
  ```

- Create or obtain the localization override files to use for each of the supported languages.

- Determine the proper location to store the localization override files accessed by the Agentry Server. The default location is the sub-folder `localizations` of the Agentry Server's installation location, though you can override this default location.
- If the single language localization method is used to provide default override values, perform that procedure first.
- Review the settings of the configuration options in the `[Configuration]` section of the `Agentry.ini` file, specifically:

```
ClientStringsFile
```

```
ApplicationStringsFile
```

```
ApplicationGlobalsFile
```

Task

1. Make a backup copy of the `Agentry.ini` configuration file and open it in a text editor.
2. Locate the section `[Configuration]` in this file. Set the value of the `localizations` option to contain a semicolon-delimited list of the locale names for the languages to support. If the `localizations` option is not listed it may be added. Following is an example of this option. It is set to provide support for two dialects of Spanish (Mexico and Span) and provide a default Spanish set of overrides: `localizations=es-ES;es-MX;es`.
3. If you are placing the localization files in a location other than the `localizations` sub-folder, set the configuration option `localizationsPath` to this location. If this option is not present, you can add it. If it is not necessary to override the default location, skip this step.
4. Verify the settings changed are accurate and close and save the `Agentry.ini` file.
5. Copy the localization override files to the `localizations` folder, or to the location specified in the `localizationsPath` configuration option and verify the names of these files match both the file name settings and the supported languages listed in the `localizations` option. Following is an example of the files needed based on support for Spanish as spoken in Spain, Mexico, and to provide a general Spanish set of overrides for other locales:

   ```
   ApplicationText.es-ES.ini ClientText.es-ES.ini
   ApplicationGlobals.es-ES.ini ApplicationText.es-MX.ini
   ClientText.es-MX.ini ApplicationGlobals.es-MX.ini
   ApplicationText.es.ini ClientText.es.ini
   ApplicationGlobals.es.ini
   ```
6. Once this procedure is complete, restart the Agentry Server.

The new override values are read in during the restart.

Once this procedure is complete, the Agentry Server uses the locale information provided by the Agentry Clients to determine the proper override values, if any, to use. Those Agentry Clients with a locale not set to one of those listed in the localizations option receive the overrides in the single language localization files; or, if these are not in use, the values as defined in the published application project are sent to Agentry Clients by the Agentry Server.

Override File Format: ClientText.ini

Shown here is an example of the contents of the `ClientText.ini` and `ClientTextBase.ini` files. This example is followed by a description of this format:

```
[Strings]

AG3_ABOUT_BMP_FILE=about.bmp

AG3_ABOUT_DIALOG_TEXT=About %1

AG3_ABOUT_MENU=&About %1...

AG3_ABOUT_TXT=about.txt

AG3_ABOUT_NAME=Agentry Client

AG3_ABOUT_OK=OK

AG3_ABOUT_PVERSION=Published As: v
```

In the above example, the first line is the section name, `[Strings]`, the only section in this file. The `ClientText` and `ClientTextBase.ini` files will be the same for all applications deployed on the same version of Agentry and for a given language. The values here are those that override built-in components of the Agentry Client and are not specific to an application.

In the above example, the items within the section are those that pertain to the *About* dialog displayed when the user clicks the **Help | About** menu item on the Client. The keys for all items begin with `AG3_[text]`. That is followed by additional text identifying the item to which the keys pertain.

The general naming convention of the keys is:

```
AG3_CATEGORY_COMPONENT
```

`CATEGORY` is the general resource, such as a built-in screen, or definition type, such as items related to actions, to which the key pertains. The `COMPONENT` is one or more words

describing the specific component of the resource identified by the `CATEGORY`. So, in the above examples, `ABOUT` refers to the *About* built-in screen, and `NAME` refers to the name value displayed in the *About* screen.

The values are the built-in display values or resources used by the Client. Changing the value overrides the item displayed on the Client. As an example, the `AG3_ABOUT_NAME` item has a value of `Agentry Client`, which is displayed in the *About* screen. This can be overridden to contain a different value, for application branding purposes, for example.

Override File Format: Globals.ini

Shown here is an example of the contents of the `Globals.ini` and `GlobalsBase.ini` files, followed by a description of this format:

```
[STRINGLENGTHS]

; AddressMax: This is the maximum length for an address value.

AddressMax=40

; AddressMin: This is the minimum length for the Address value.

AddressMin=1

; CityMax: This is the maximum length for the City value.

CityMax=12

; CityMin: This is the minimum length for the City value.

CityMin=1

; PhoneMax: This is the maximum length for the Phone value.

PhoneMax=22

; PhoneMin: This is the minimum length for the Phone value.

PhoneMin=1
```

The first line of this example is the section name `[STRINGLENGTHS]`. This section name corresponds to a defined global group within the application project. Each global group has a

section within this file. All globals defined within a group are listed within the appropriate section.

The second line in the example is a comment line. All comments within this file are preceded by a semicolon. Comments in this file are automatically generated by the Editor when the file is created. The comments are the description values of each global definition. You can also add comments to the base file and override file as needed.

The third line, `AddressMax=40`, is the first global definition within the `StringLengths` group. The name of the global to which this setting pertains is `AddressMax`. Its defined value within the application project is 40.

Changing this value overrides the defined behavior for the application. More specifically, the value entered in the override file is the one for the global on the Client, affecting any other definitions that reference the global.

Override File Format: ApplicationText.ini

Shown here is an example of the contents of the `ApplicationText.ini` and `ApplicationTextBase.ini` files, followed by a description of this format.

```
[Customers]

module=Customers

object.MainObject=Main Object

property.MainObject.Customers=Customers

object.Customer=Customer

property.Customer.CustomerID=ID

property.Customer.CompanyName=Company

property.Customer.ContactName=Contact
```

Note: The contents of this file are application-specific and will differ from one application to the next. The above is an example for a mobile application built for the Northwind demonstration database in SQL Server.

This override file contains one section for each module defined within the application, named `[`*`ModuleName`*`]`, and one section named `[Misc.]` for the application-level definitions. Any items listed under a section name are part of that section.

The second line in the above example, `module=Customers`, represents the display name of the module. Each override item's key follows a defined format:

```
defType.ModuleLevelParent.parent.parent.definitionName
```

The following items are the explanations to each component of the key:

- `defType`: The type of definition that is overridden by the item. Examples include:
  - `property`
  - `object`
  - `platform`
  - `listscreencaption`
- `ModuleLevelParent`: The name of the module-level parent definition of the definition whose display value is overridden.
- `Parent.Parent...`: The name of each ancestor definition between the module level ancestor and the definition that is overridden.
- `definitionName`: The name of the definition that is overridden.

The last line in the above example, `property.Customer.ContactName`, contains the override value for the display name of the `property` in the `Customer` object named `ContactName`. All definitions within the application with a display name, label, or caption attribute are listed in this file.

Override File Format: Strings.properties

Shown here is an example of the contents of the `strings.properties` file. This example is followed by a description of this format:

```
java.text.creatingSession=%s - Creating session to host %s for
user %s... java.text.inventoryItemNotFound=Item number %s not
found in storeroom %s in site %s. java.text.noAppSecurity=%s
does not have security rights in site %s to app %s.
```

The values here are those that override strings returned from the Java back end. The keys for all items begin with java.text. That is followed by the message name and an equals sign. To the right of the equals sign, =, is the message text that will be displayed. The message text may contain variables which begin with a percent sign, %. All message text, except for variables, should be translated to the required language.

```
java.text.creatingSession=%s - Creating session to host %s for
user %s...
```

In the example above, the `%s` variable appears three times in the message text and must not be modified. The text "Creating session to host" and "for user" may be localized.

CHAPTER 6 Data Tier Administration

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

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

Note: All data tier nodes in a cluster must run with the same timezone setting; otherwise, data issues may occur.

Changing Database Ports for SQLAnywhere Databases

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

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

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

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

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

   For a list of database services, search for *SAP Mobile Platform Windows Services* in *System Administration*.
2. If you are updating ports in a cluster deployment, then for each database that requires a port change, open the corresponding initialization file and modify the `tcpip(PORT=newPort)` entry. These files are installed to *`SMP_HOME`*`\Servers\SQLAnywhere12\bin`*`xx`*:

   - For the cache database, open `cdboptions.ini`.
   - For the cluster database, open `cldboptions.ini`.
   - For the log database, open `monitoroptions.ini`.

3. Make corresponding port number changes for eace datasource configuration files, which is located in *SMP_HOME*\Servers\UnwiredServer\Repository\Instance\com\sybase\djc\sql\DataSource.

   For example, `default.properties` for default database, and `clusterdb.properties` for the cluster database.
4. Restart the cluster database.
5. For all deployments, run the update properties utility to propagate changes to all runtime servers:

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

   Supported server port property names include:

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

   For details on the update properties utility, see *Update Properties (updateprops.bat) Utility* in the *System Administration* guide.
6. Restart all SAP Mobile Servers.

See also

- *Changing SQLAnywhere Database Server Startup Options* on page 64
- *Update Properties (updateprops.bat) Utility* on page 284
- *Cache Database Startup Options (cdboptions.ini) Initialization File* on page 337

Changing SQLAnywhere Database Server Startup Options

Change the database server startup options (thread counts and user options) to control the performance of a SQLAnywhere database server.

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

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

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

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

2. If you are thread count and user startup options in a cluster deployment, then for each database that requires a startup option change, open the corresponding initialization file. These files are installed to *SMP_HOME*\Servers\SQLAnywhere12\bin*xx*:
   - For the cache database, open `cdboptions.ini`.
   - For the cluster database, open `cldboptions.ini`.
   - For the log database, open `monitoroptions.ini`.
3. Modify one of the startup options in the `.ini` files, for example:

   ```
   -gn <threadcount> -c <cachesize>
   ```
4. Ensure all the database services are restarted.
5. Restart all SAP Mobile Servers.

See also

- *Changing Database Ports for SQLAnywhere Databases* on page 63
- *Update Properties (updateprops.bat) Utility* on page 284
- *Cache Database Startup Options (cdboptions.ini) Initialization File* on page 337

Using a Different Database Log Path

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

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

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

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

   For example:

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

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

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

Setting Up an Existing Database for Monitoring

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

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

1. Use dbisql to run the SQL scripts that set up the monitoring schema. This utility is located in *SMP_HOME*\Servers\SQLAnywhere*XX*\BIN*XX*.
   - `init-monitoring-tables-asa.sql` – sets up the SQL Anywhere database with a general monitoring schema.
   - `ASA_MONITORING_DDL.sql` – sets up the SQL Anywhere database with a cache monitoring schema.

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

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

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

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

   This installs the monitor and domain log database as a service (named SAP Mobile Platform LogDataDB Service).
2. Disable the cache, cluster and messaging database services.
3. Connect to SAP Mobile Server cluster from SAP Control Center and:
   a) In the navigation pane, expand the **default** domain node, then click **Connections**.
   b) Click the **Connections** tab.
   c) For each connection (monitordb and domainlogdb), click **Properties**, then update the host and port values to reflect the connection properties of the new host node.
   d) Click Save.

See also

- *Planning for Domain Logging* on page 179

Managing Connection Pools for SAP Mobile Server Connections

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

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

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

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

See also

- *JDBC Properties* on page 240

Rebuilding Platform Databases

Platform databases need to be rebuilt regularly. Without regular maintenance, the database log can grow to be large (several gigabytes in size), and performance could degrade.

If you experience long shutdown times with data services for the data tier, you need to unload and reload data the dbunload utility. This rebuilds your database and maintains a healthy data tier performance.

To avoid this issue, SAP recommends that you perform this action every four or five months.

1. Stop all runtime services (data tier and server nodes), including the cache database, the cluster database, and the messaging database, and synchronization services.
2. Perform a full off-line backup of each, by copying the database and transaction log files to a secure location. See *Backup and Recovery*.
3. Rebuild each runtime database with the SQL Anywhere Unload (dbunload) utility.

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

Control Transaction Log Size

Control the size of transaction logs to prevent the log files from growing indefinitely.

Use the SQL Anywhere **dbbackup** utility, with the **-xo** flag. The **-xo** flag deletes the current transaction log file, once it has been backed up successfully, and creates a new one. See *SQL Anywhere® Server – Database Administration* for information.

Alternately, use a variant of the SQL Anywhere **BACKUP DATABASE** command. This example performs daily backups automatically from within the database server:

```
CREATE EVENT NightlyBackup
SCHEDULE
START TIME '23:00' EVERY 24 HOURS
HANDLER
BEGIN
   DECLARE dest LONG VARCHAR;
   DECLARE day_name CHAR(20);

   SET day_name = DATENAME( WEEKDAY, CURRENT DATE );
   SET dest = 'd:\\backups\\' || day_name;
            BACKUP DATABASE DIRECTORY dest
   TRANSACTION LOG RENAME;
END;
```

CHAPTER 7 EIS Connection Management

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

EIS connections include:

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

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

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

See also

- *EIS Data Source Connection Properties Reference* on page 240
- *Creating Data Source Connections for a Domain* on page 79

Data Source Connections

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

For SAP Mobile Server to recognize a EIS data source, you must define a connection to that data repository. The connections are defined in SAP Control Center with the SAP Mobile

Platform perspective, and are known as server-to-server connections because they are opened by SAP Mobile Server.

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

See also

- *JCo Connection Rules* on page 71

Connection Templates

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

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

Changing Connections to Production Data Sources

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

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

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

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

   The client user credentials are used to establish connection with the back-end. Administrators should determine from their development team which MBOs are affected. You must still remap connections to production values.

3. (Optional) Tune properties (such as connection pool size), to improve the performance of production-ready applications.

Viewing and Editing EIS Connection Properties

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

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

See also

- *EIS Data Source Connection Properties Reference* on page 240

Tuning Connection Pool Sizes

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

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

See also

- *EIS Connection Performance Tuning Reference* on page 95

JCo Connection Rules

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

By default, SAP Mobile Platform JCo connections follow these rules:

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

See also

- *Data Source Connections* on page 69

CHAPTER 8 Domain Management

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

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

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

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

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

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

Table 3. Domain management tasks

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

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

Enabling a Multitenancy Environment with Domains

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

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

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

1. *Determining a Tenancy Strategy*

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

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

   Define a set of security providers in SAP Control Center to protect domain resources, according to the specific security requirements of the domain. Create a security configuration, then map it to the desired domain.
4. *Activating a Domain Administrator*

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

5. *Assigning Domain Administrators to a Domain*

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

6. *Mapping Roles for a Domain*

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

7. *Creating Data Source Connections for a Domain*

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

Determining a Tenancy Strategy

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

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

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

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

configurations for each of these packages. This option requires more administrative time than grouping all packages belonging to the same EIS into one domain.

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

Creating and Enabling a New Domain

Create and configure multiple domains within a single SAP Mobile 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 SAP Control Center.
2. In the left navigation pane, select the **Domains** folder.
3. In the right administration pane, select the **General** tab, and click **New**.
4. In the Create Domain dialog, enter a name for the domain and click **Next**.

   Note: Domain names are case insensitive.
5. Select a security configuration for the domain by checking an option from the list of available configurations. You must select at least one security configuration. The security configurations you select are then available for use in validating users accessing the packages. If you select multiple security configurations, the first one you select becomes the default security configuration for the domain.
6. Click **Next**.
7. Optional. Select one or more domain administrators for the domain.
8. Click **Finish**.
   The new domain appears in the **General** tab.
9. Click the box adjacent to the domain name, click **Enable**, then click **Yes** to confirm.

Creating a Security Configuration for a Domain

Define a set of security providers in SAP Control Center to protect domain resources, according to the specific security requirements of the domain. Create a security configuration, then map it to the desired domain.

A security configuration determines the scope of data access and security. A user must be part of the security repository used by the configured security providers to access any resources

(that is, either a SAP Control Center administration feature or a data set from a back-end data source) on a domain. See *Security Configurations* in the *Security* guide.

When you create a new security configuration, SAP Mobile Platform sets the NoSecurity provider by default. SAP recommends that after you add, configure, and validate your providers, you remove the NoSecurity provider. For more information on the NoSecurity provider, see *NoSecurity Configuration Properties* in *SAP Control Center for SAP Mobile Platform*.

1. In SAP Control Center, add a new security configuration using the **Security** node.
2. In the left navigation pane, expand the **Security** folder and select the new configuration.
3. Use the **Authentication**, **Authorization**, **Attribution**, and **Audit** tabs to configure the appropriate security providers for each aspect of domain security.
4. Edit the security provider properties, as required.
5. Validate the configuration to ensure that SAP Mobile Server accepts the changes.
6. Apply the changes to SAP Mobile Server.

Activating a Domain Administrator

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

Prerequisites

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

Task

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

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

Assigning Domain Administrators to a Domain

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

Prerequisites

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

Task

1. Open SAP Control Center.
2. In the left navigation pane, expand the **Domains** folder, and select the domain for which to assign domain administration privileges.
3. Select the domain-level **Security** folder.
4. In the right administration pane, select the **Domain Administrators** tab, and click **Assign**.
5. Select one or more administrator users to assign to the domain by checking the box adjacent to the user name.
6. Click **OK**.
   A message appears above the right administration pane menu indicating the success or failure of the assignment. If successful, the new domain administrator appears in the list of users.

Mapping Roles for a Domain

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

Prerequisites

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

Task

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

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

Creating Data Source Connections for a Domain

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

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

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

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

See also

- *Chapter 7, EIS Connection Management* on page 69

Scheduling Accumulated Data Cleanup for Domains

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

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

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

1. In the SAP Control Center left navigation pane, expand the **Domains** tab and select a domain.
2. In the right pane, select the **Scheduled Task** tab.
3. Under Task, select one of the options you want to schedule, and then select **Properties** to set up its automatic schedule:

Option	Description
Subscription Cleanup	Removes subscriptions that are not active for the 'number of inactive days' in the schedule task configuration. Note, subscription is considered active as follows: • Replication – last synchronization request time-stamp. • Messaging – last synchronization message time-stamp. **Note:** If a casual user accesses the system infrequently, for example three to four times a year, the user falls outside the specified time frame and is removed from SAP Mobile Platform. The user will then have to reinstall the application and initiate a sync to re-activate subscription.
Error History Cleanup	Removes historical data on MBO data refresh and operation replay failures, which result from system or application failures. System failures may include network problems, credential issues, and back-end system failure. Application failures may include invalid values, and non-unique data. **Note:** Only error messages are removed.
Client Log Cleanup	Removes client log records that have already been synchronized to the device, or are no longer associated with active users.
Synchronization Cache Cleanup	This cleanup task removes: • Logically deleted rows in the cache that are older than the oldest synchronization time on record in the system. Synchronization activity for all clients establish the oldest synchronization time. • Unused or stale partitions. In addition, the optional **exp** header property in DCN messages indicates an expiration date, after which the data is purged based on the defined Synchronization Cache Cleanup schedule.

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

CHAPTER 9 Cache Data Management

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

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

- Data change notifications (DCNs) -- when data used by an application changes on an EIS server, DCNs notify SAP Mobile Server to synchronize cache data. DCNs are useful to cache management, since they facilitate time-sensitive data synchronization between scheduled cache refreshes.
- Cache refresh schedules -- update the cache with the most recent EIS data at regularly scheduled intervals, or on demand. The remote client database eventually retrieves updated data from the server's local copy in the CDB, using one of the supported synchronization triggers. Cache refresh schedules ensure data availability while managing the network traffic required to maintain that data.

Data Change Notifications

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

Developers typically use DCNs because they:

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

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

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

- Users do not always know when to synchronize. Waiting for a user-triggered synchronization is unreliable.

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

You can configure either secure (HTTPS) or non-secure (HTTP) for DCNs. For details on how DCNs are created and sent see the *Mobile Data Models: Using Mobile Business Objects*.

Cache Refreshes

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

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

Scheduled cache refreshes control the frequency with which objects update data and regulate network traffic by synchronizing data at strategic times, rather than pushing data changes through as they occur. Administrators can also perform on-demand cache refreshes for cache groups at a specified time.

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

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

Example Data Update Models

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

Scenario 1: Product Sales with Expected and Unexpected Changes

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

To support this deployment the administrator might:

1. Evaluate the business context.

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

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

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

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

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

Scenario 2: Urgent Alerts using Subscriptions and Schedules

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

To support this deployment, an administrator might:

1. Evaluate the business context.

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

   The SAP Mobile Server administrator responds in such a way that when a new alert is entered into the back-end server, the alert is delivered to the corresponding doctor by setting the cache interval to 0 or "real time". Data is always live and immediate with no caching involved. However, the administrator also sets the schedule repeat for 1 minute

and is required 24 hours a day, seven days a week. This refresh schedule is paired with a subscription template for the UrgentAlert MBO with push synchronization enabled.

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

CHAPTER 10 Performance Tuning and Testing

Whether you use the replication or messaging payload protocol, tuning synchronization provides the highest throughput while limiting CPU and memory consumption to reasonable operational levels. You can refine performance an ongoing capacity, or when new applications are deployed, or if there are changes to existing application functionality, load, and so on. Load testing measures performance under simulated use.

Tuning recommendations vary depending the payload protocol.

Performance Considerations for Relay Server and Outbound Enabler

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

Table 4. Recommendations for 64-bit Production Servers

Component	Function	Default	Production Recommendation
Relay Server shared memory	Settings for shared memory buffer. Remember that shared memory must account for concurrent connections, especially with large payloads.	10MB	2GB

Component	Function	Default	Production Recommendation
Relay Server response times	To decrease the small payload response time of a Relay Server that is lightly loaded, an administrator can add `up_pad_size=0` to the Relay Server configuration file created from SAP Control Center. Adding this value may have adverse affects for a highly loaded server since this setting defeats the built-in network Nagle's algorithm, which delays sending small packets in order to improve overall throughput for many small packets.	None	0
Outbound Enabler deployment	With SQL Anywhere®, each RSOE provides an upload and download channel to the relay server.	1 Outbound Enabler per server	3 Outbound Enablers per synchronization (replication or messaging) port, especially when initial synchronizations are larger than 4MB

See also

- *Performance Considerations for Replication* on page 87
- *Performance Considerations for Messaging* on page 96
- *LoadRunner Extension for SAP Mobile Platform* on page 103
- *Enabling RBS Performance Testing with LoadRunner* on page 109

Performance Considerations for Replication

Understand fundamentals of entity-state replication, so you can understand how to improve performance for this protocol.

Entity-state replication has two distinct operation and data transfer phases where only differences are transferred: upload (where only updates from the mobile client to the server are integrated with data in the mobile middleware cache and pushed to the EIS) and download (where EIS delta changes are determined for, and supplied to, the specific client). These phases are conducted as discrete transactions to protect the integrity of synchronization. Synchronizations are carried out within a single data transfer session, allowing for exchanges of large payloads. Therefore the goal of tuning performance for the replication payload is to ensure those transfers of large volumes are handled effectively and not to create bottlenecks in the runtime.

Considerations that affect performance can be separated into synchronization phases and architecture components.

Entity-state replication use cases center around three primary phases in moving data between server and client, each of which need to be considered when choosing performance setting values. Each phase describes mobility environment components, and how they affect performance: MBO development and data design, EIS data models , and SAP Mobile Server runtimes. These phases are:

- Initial synchronization – where data is moved from the back-end enterprise system, through the mobile middleware, and finally onto the device storage. This phase represents the point where a new device is put into service or data is reinitialized, and therefore represents the largest movement of data of all the scenarios. In these performance test scenarios, the device-side file may be purged to represent a fresh synchronization, or preserved to represent specific cache refreshes on the server.
  For this phase, the most important performance consideration is the data and how it's partitioned (EIS design), and loaded by the MBO (MBO development) using operation parameters, synchronization filter parameters). Synchronization parameters determine what data is relevant to a device; they are used as filters within the server-side cache. Load parameters determine what data to load from the EIS into the cache.
- Incremental synchronization – involves create, update, and delete (CUD) operations on the device where some data is already populated on the device and changes are made to that data which then need to be reconciled with the cache and the back-end enterprise system. When create and update operations occur, changes may be pushed through the cache to the back end, reads may occur to properly represent the system of record, for example, data on the back end may be reformatted, or both. This scenario represents incremental changes to and from the system of record.
  As in the initial synchronization phase, the EIS accounts for the bulk of the device synchronization response time: a slow EIS consumes resources in the SAP Mobile Server, with the potential to further impede devices that are competing for resources in connection

pools. Additionally, the complexity of the mobile model, measured by the number of relationships between MBOs, has a significant impact on create, update, and delete operation performance. Shared partitions among users or complex locking scenarios involving EIS operations can become a major performance concern during device update operations. Cache and EIS updates are accomplished within the scope of a single transaction, so other users in the same partition are locked out during an update of that partition. Consider denormalizing the model if complex relationships cause performance concerns.

- Data change notification (DCN) – changes to the back-end data are pushed to the mobile middleware and then reconciled with the device data. DCN is typically observed in the context of additional changes to the device data so that changes from both device and back end are simultaneously impacting the mobile middleware cache.

  DCN efficiently updates the SAP Mobile Server because it does not require the SAP Mobile Server to poll the EIS or to refresh the cache based on a schedule. EIS DCN applied to the cache is independent of the client synchronizations. If DCN data is located in a shared partition, multiple devices benefit from the single EIS update to the cache. There are several ways to materially improve DCN performance:
  - Use a load-balancer between the EIS and the cache – DCNs can be efficiently applied across a cluster, as each node in the cluster helps to parse incoming payloads.
  - Combine multiple updates into a single batch.
  - Run DCNs from a multithreaded source to parallelize updates. Note that there is a diminishing return beyond three to four clients, in large part due to the nature of the model.

Different models exhibit different performance characteristics when applying updates, so proper analysis of application behavior is important.

See also

- *Performance Considerations for Relay Server and Outbound Enabler* on page 85
- *Performance Considerations for Messaging* on page 96
- *LoadRunner Extension for SAP Mobile Platform* on page 103
- *Enabling RBS Performance Testing with LoadRunner* on page 109

Overview of Replication Tuning Recommendations

Replication tuning recommendations are designed to maximize bandwidth between the Relay Server and its Outbound Enablers, SAP Mobile Server, and the EIS connections.

SAP recommendations touch on components outside and inside the LAN:

- For large replication payloads greater than 4MB, increase the bandwidth between the Relay Server and the SAP Mobile Platform cluster nodes by increasing the number of Relay Server Outbound Enablers (RSOEs) and increasing the shared memory of the relay servers.

- Ensure adequate processing bandwidth for peak conditions by setting an adequate replication thread count for the entire cluster (the combination of each node's replication thread count) and JVM memory settings you configure for SAP Mobile Server. The replication thread count is the point where SAP Mobile Server and cache database (CDB) throttling, or limiting, occurs.

  The speed of the CDB storage and storage controllers is the single most important factor in providing good system performance. Staging of mobile data is performed within the CDB in a persistent manner such that if a device synchronizes. The CDB database server itself is largely self-tuning, although typical database maintenance is essential for proper performance.
- Provide as many resources as are available as you work back to the EIS. Do not limit connection pools.

SAP Mobile Server Replication Tuning Reference

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

Table 5. Recommendations Generally Suitable for 64-bit Production Servers

Property	Function	Default	Production Recommendation
Thread count	Controls the concurrent number of threads that service devices for synchronization. This setting controls the amount of CPU used by the SAP Mobile Platform and CDB tiers. If the processor of either the CDB or the SAP Mobile Server is excessively high, you can use this setting to throttle the number of requests and limit contention for resources. Low settings decrease parallelism; high settings may cause undue contention for resources.	10 per server	For a 2-node cluster, use 12 for each node. For a single node, use 24.

Property	Function	Default	Production Recommendation
Synchronization cache size	The maximum amount of memory the server uses for holding table data related to device users, network buffers, cached download data, and other structures used for synchronization. When the server has more data than can be held in this memory pool, the data is stored on disk.	70p where p is is a percentage either of the physical system memory, or of the process addressable space, whichever is lower.	On 64-bit database servers, the cache size can be considered unlimited .
JVM minimum heap size	The minimum memory allocated to the differencing and cache management functions of the server.	512MB	
JVM maximum heap size	The maximum memory allocated to the differencing and cache management functions of the server. Choose the heap size carefully, otherwise you may have issues with SAP Mobile Server startup. Choose a JVM setting that is appropriate for your: • Server hardware configuration (for example, 64 bit/32 bit, RAM size, VM size). • Size of objects and encoding. Because SAP Mobile Server uses base64 as default binary encoding, it causes a 3-factor growth from the original size. For example, 1M after encoding to base64, uses around 3M memory. For multiple concurrent users, the memory would be multiplied. So we need to calculate the maximum heap size based of server based on above information.	2048MB	For a 64-bit operating system, SAP recommends 1 gigabyte for a normal configuration, but 2 gigabyte for a stress configuration (which can vary depending on what RAM is available).

Note: The synchronization differencing algorithms are a key feature of replication; this technology runs in the JVM. You must provide adequate memory to these components. If

these algorithms are memory starved, the JVM spends an inordinate amount of time garbage collecting memory, and synchronizations back up in the internal queues. You can monitor process memory usage with tools like SysInternal's Process Explorer to determine the actual amount of memory in use by SAP Mobile Platform, and adjust the JVM heap size accordingly

Tuning Synchronization for Replication Payloads

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

1. Isolate the monitoring and domain logging databases from the cache server.

   This isolation only needed if you are using monitoring and domain logging in the production system, and want to reduce any database or storage contention. See *Isolating the Monitoring and Domain Logging Databases from Cache and Messaging Databases*.
2. Disable all enabled monitoring profiles that currently exist.

   Normally, monitoring in a cluster configuration occurs at two levels — the cluster and the domain. SAP recommends that you turn off monitoring when assessing performance.

   a) In navigation pane of SAP Control Center, click **Monitoring**.
   b) In the administration pane, select the profile name and click **Disable**.
   c) Validate that all monitoring is disabled by opening *SMP_HOME*`\Servers \UnwiredServer\Repository\Instance\com\sybase\sup\server \monitoring\ MonitoringConfiguration \domainLogging.properties`, and verifying that "status", "enabled", and "autoStart" are set to false.

   See *Monitoring Profiles* in the *SAP Control Center for SAP Mobile Platform* online help.
3. In SAP Control Center, stop all SAP Mobile Servers in all clusters.
4. Stop the cache service on the data tier node.
5. Use the dblog utility to isolate the cache disk and log on to fast storage.

   For details, see *Using a Different Database and Log Path*.
6. Set new cache thread count, pool size, and initial cache size values.

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

   a) In the left navigation pane, click **Configuration**.
   b) In the right administration pane, click the **General** tab.
   c) To configure replication ports and optional properties, select **Components** from the menu bar, select **Replication**, and click **Properties**.

d) To configure **Synchronization Cache Size** and **Thread Count**, select **Performance** from the menu bar, For production recommendations on the values for these properties, see *SAP Mobile Server Tuning Reference* in *System Administration*.

9. In SAP Control Center, set thread stack size and JVM heap sizes for the server.

a) In the left navigation pane, click **Servers**>**SeverName**>**Server Configuration**.
b) In the right administration pane, click the **General** tab.
c) Adjust the thread stack size and JVM heap sizes to the new values.
   - Thread Stack Size – the JVM `-Xss` option.
   - Minimum Heap Size – the minimum size of the JVM memory allocation pool, in megabytes. For production recommendations on this value, see *SAP Mobile Server Replication Tuning Reference* in *System Administration*.
   - Maximum Heap Size – the maximum size of the JVM memory allocation pool. Use K to indicate kilobytes, M to indicate megabytes, or G to indicate gigabytes. For production recommendations on this value, see *SAP Mobile Server Replication Tuning Reference* in *System Administration*.
   - User Options (in Show optional properties) – other JVM options.
d) Repeat for all servers.

10. Restart all SAP Mobile Servers.

Testing CPU Loads for Replication

Perform a mixed-load test to ascertain whether the throughput to the cache and EIS is sufficient. Do this by testing then observing CPU load trends.

1. Run a typical application maximum mixed-load test on the SAP Mobile Server (5% initial sync, 95% subsequent sync) and observe the cache CPU load.

   Do not run this mixed-load test against Relay Server; you want to first ascertain the maximum throughput of just the SAP Mobile Server and cache.

2. If the cache CPU load is too high, too many threads could be impacting performance:

a) Decrease the replication thread count on each server in SAP Control Center.
b) Restart all servers.
c) Observe the cache CPU load again.

3. If the cache CPU load is low, fine tune the replication thread count until one yields the maximum through put. Check this by:

a) Note the client response times of other synchronization types and try increasing the replication thread count in small increments.

   Other types of synchronization include initial to initial, subsequent to subsequent, similar payload, cache policy, and so on.

b) Repeat this process until the synchronization response times are as low as possible for the same number of clients for all types of synchronizations.

   Once the relative client response time is as low as possible without further increasing the replication thread count, you have reached the configuration that yields the

maximum throughput. In other words, tune the thread count to yield the best overall response times. Usually, this thread count number is small.

4. Repeat the mixed-load test on EIS backends.

   This checks for EIS latencies and cache policies, both of which can change the performance of the server cluster.

5. Introduce relay server into the environment, and repeat the mixed-load test.
6. Once the environment has stabilized and tuned, enable on and configure only the monitoring profiles you need.

See also

- *Configuring SAP Mobile Server General Properties* on page 20

Cache Database Performance Tuning Reference

Use this table to quickly ascertain which cache database (CDB) properties you can adjust to tune performance.

Because the replication payload protocol uses entity-state replication and a differencing algorithm to determine what to synchronize to each device, the CDB is one of the most critical components for performance in terms of processing power, memory, and disk performance. The CDB must also be scaled vertically (on larger hardware) because it supports all of the nodes of the cluster.

Table 6. Recommendations Generally Suitable for 64-bit Production Servers

Property	Function	Default	Production Recommendation
Thread count	Limits the number of tasks (both user and system requests) that the database server can execute concurrently. If the database server receives an additional request while at this limit, the new request must wait until an executing task completes.	20	200
Initial cache size	Sets the initial memory reserved for caching database pages and other server information. The more cache memory that can be given the server, the better its performance.	24MB	8GB

Property	Function	Default	Production Recommendation
Disk configuration	Isolate the data and log on distinct physical disks. Of the two files, the log receives the most activity. Use disk controllers and SAN infrastructure with write-ahead caching in addition to high-speed disk spindles or static memory disks. The same database must support all cluster members. The faster these database drives perform, the better the performance on the entire cluster.	1 disk	2 disks 10K RPM
Connection pool maximum size	Sets the maximum (JDBC) connection pool size for connecting to the CDB from each cluster member. In the example production configuration, each SAP Mobile Server is allowed more connections than the number of threads set in the CDB thread configuration. This ratio helps to ensure that the CDB database server is the control point for limiting resource contention in the cluster. In a server configured with the CDB connection pool settings set to 0, the actual number of connections used will correspond to the number of SAP Mobile Server threads in use. The number of internal connections in use at any one time per thread varies, although fewer than four is typical.	100	0 (unlimited)

See also

- *Configuring Initial Cache Size for Performance* on page 94

Configuring Initial Cache Size for Performance

Configure the initial memory reserved for caching database pages to enhance performance.

1. Stop SAP Mobile Server.
   See *Starting and Stopping SAP Mobile Server.*
2. Shutdown and remove the consolidated database (CDB) service using the batch file in *SMP_HOME*`\Servers\UnwiredServer\bin`.

```
asadbservice.bat stop
```

```
asadbservice.bat remove
```

3. Set SAP Mobile Platform services to manual start if they are designated as automatic start.
4. Restart SAP Mobile Server.
5. Modify *SMP_HOME*\Servers\UnwiredServer\config\configure-sup.xmlas follows:

   Change `-c 24M` to `-c 5G`.
6. Reinstall the CDB service.

   ```
   asadbservice.bat install manual
   ```

   or

   ```
   asadbservice.bat install auto
   ```
7. Set SAP Mobile Platform services to automatic start.
8. Restart SAP Mobile Server.

See also

- *Cache Database Performance Tuning Reference* on page 93

EIS Connection Performance Tuning Reference

Use this topic to quickly ascertain which EIS connectivity properties you can adjust to tune performance.

Property	Function	Default	Production Recommendation
Connection pool maximum size	Ensure that adequate EIS resources are allocated for servicing the mobile infrastructure. The actual number of connections necessary varies, based on the maximum number of SAP Mobile Platform threads in use at any time and the duration it takes for the EIS to respond. If possible, allocate a connection for each thread (or leave the setting unbounded). If you must limit the number of EIS connections in the pool to a lower number than the number of SAP Mobile Platform threads and you experience timeouts, you may need to adjust the EIS connection timeout values ; however, these connections will impede other threads competing for EIS connections.	Varies, depending on type: JDBC is 10, Proxy is 25, and Web Services, SAP DOE, and SAP JCo have no limit by default.	0 (unlimited)

See also

- *Tuning Connection Pool Sizes* on page 71

Performance Considerations for Messaging

Understand fundamentals of messaging so you can understand how to improve performance for this payload protocol.

With messaging payloads there is a trade off between efficiency and immediacy. Some messages arrive earlier than in entity-state replication because the payload is spread out over many relatively small messages that are delivered individually, as opposed to being delivered as a single large download. Therefore the goal of tuning messaging performance is to perform multiple transfers of small payload volumes quickly, in addition to keeping changes in the correct order.

The messaging use cases center around three primary phases in moving data between server and client, each of which need to be considered when choosing performance setting values. Each phase describes mobility environment components, and how they affect performance: MBO development and data design, EIS data models and runtime servers, and SAP Mobile Server runtimes. These phases are:

- Initial subscription – A mobile application subscribes to an messaging package to receive data as "import" messages from the server. Upon the receipt of the subscription request from the device, the server checks security and executes a query against the cache database (CDB) to retrieve the data set, which it then turns into a series of import messages to be sent to the device. The maximum message size is currently fixed at 20KB. Increasing the maximum allowable size would allow the same amount of import data with fewer messages. However, in some devices, large message size can trigger resource exhaustion and failures.

  The subscription phase is the most expensive or time consuming portion of the life cycle, especially for a large initial data set. However, compared to entity-state replication payloads, messaging payloads can take longer to populate the data on the device database because of the aggregation of fine-grained messages. This latency is due to the additional cost of providing reliability to the delivery of every message. In addition, a significant amount of processing is incurred on both the server and the client side to marshal messages. While this marshaling may not necessarily impact the server, it places a heavy burden on the device, especially if the device is on the low end of the performance spectrum. The message import is also limited by the device's nonvolatile storage write performance.
- Subsequent Synchronization – Device-side data changes due to the user interacting with a mobile application. It is important that you understand the unit of change. An operation with an associated tree of objects with a containment (or composite) relationship is considered a unit of change. Changes are wrapped in a message with the appropriate operation type: create or update. (The delete operation requires only the primary key that identifies the root of the tree to be transmitted, rather than the entire object tree). Upon

receipt of the message, the server replays the operation against the EIS and returns a replay result message with one or more import messages that reflect the new state of the data. Unlike replication, the unit of change is pushed to the device as soon as the changes are ready, so subsequent synchronization is occurring in the form of many messages. As a result, the synchronization happens over time as a stream of messages.

The subsequent synchronization phase addresses mobile devices sending CUD requests to SAP Mobile Platform and receiving responses and updates as a result of the operations. In general, the limiting factor for performance is on the EIS as the CUD requests are replayed. You may experience SAP Mobile Server performance issues if requests are spread over a number of back-end systems. If the EIS response time is large, you may need to allocate additional threads to replay requests concurrently against the EIS.

- Incremental Synchronization – The updates sent to the device as messages due to DCN from the EIS. DCN is the most efficient way for a back-end system to notify SAP Mobile Platform of changes to data mobilized to the device (as opposed to polling for EIS changes).Because messaging uses push synchronization to send out the updates as import messages to the devices, each DCN normally causes updates to device data over the messaging channels, based on a configurable schedule within the server. The main reason for this behavior is the concern for activity storms arising from batched DCNs where many granular changes on the cache cause flurries of messages that might be better consolidated first on the server. Currently, most EIS back-ends are not event-driven and DCNs are batched. Hence, having batched triggers directly driving an event-based model can decrease efficiency without increasing data freshness.

See also

- *Performance Considerations for Relay Server and Outbound Enabler* on page 85
- *Performance Considerations for Replication* on page 87
- *LoadRunner Extension for SAP Mobile Platform* on page 103
- *Enabling RBS Performance Testing with LoadRunner* on page 109

Overview of Messaging Performance Recommendations

Messaging tuning recommendations are designed to maximize throughput of messages, as there are unique implications to using messages as the medium of data exchange. The messaging processing limit (throttle) is determined by the number of inbound message queues. Any SAP Mobile Server within a cluster can process messages on inbound queues. The number of inbound queues is equal to the number of parallel messaging package requests that the cluster can concurrently service.

SAP recommendations touch on components outside and inside the LAN:

- Add multiple server nodes to increase data marshaling. Testing has indicated that one SAP Mobile Server node is capable of sustaining around 70 messages per second to devices. A second SAP Mobile Server node provides an additional 60% increase in delivery

capability. This increase is because the second messaging node increases message marshaling and transmission capability for the entire device population.

- Device performance and number of devices deploy. Device processing capacity limits the capacity of each message to 20KB. Due to serialization of data to JSON, a roughly 2X increase in size is observed within our data model, i.e. 4MB worth of data is represented by 9MB of serialized information for packaging into messages. As a result, the number of import messages is in the low 400s. The 2X factor is only an estimate for a moderately complex application; the true cost depends on the number of attributes (and the datatypes) of each MBO in the model. Furthermore, if push messages are to be generated for a device, you should tune the environment after executing download SQL queries against the cache: the cost of the query depends on the complexity of the download logic (number of MBO relationships) in addition to synchronization timestamp comparison. For a large number of devices, this cost can be considerable.
- Approach initial synchronization carefully, and follow a sound rollout policy for new mobile applications. The easiest approach is to spread the rollout over a period a time to avoid a large number of devices attempting to download large amounts of data. Excessive load not only slows the rollout, but may also impact performance for existing messaging applications. Apart from SAP Mobile Platform capacity and connectivity bandwidth, the actual time required to perform the initial synchronization depends mostly on the capability of the device.
- Allocate additional threads to replay requests concurrently against the EIS. Choosing a value depends upon:
  - The number of concurrent CUD operations that an EIS can handle without causing degradation
  - The number of simultaneously active EIS operations the load is spread across
  - The average response time for various EIS operations
  - Any lock contention points in the mobile model that exist due to shared data partitions
  - The total application landscape (SAP Mobile Platform does not assign threads on a per package basis. All threads are available to service CUD operations for any of the deployed messaging packages.)
- The SAP Mobile Server data change check frequency for each package. The higher the check frequency, the higher the cost, but data freshness may not actually increase. The gating factor is the frequency of updates from the EIS. If the cache is configured to refresh at midnight, after the EIS performs certain batch processing, it is most efficient to configure the scheduler to check for changes sometime after the cache refresh is completed. For an EIS that uses DCN to update the cache, the amount and frequency of changes depends on the business process. Understanding the data flow pattern from EIS to cache is crucial to determine how frequently the SAP Mobile Server is to check for changes to be pushed to the devices.
- Batch multiple DCN updates in a single notification to reduce overhead. However, a batch that is too large may impact device synchronization response due to contention. A storm of DCNs can create significant work for devices. The tradeoff is efficiency versus performance (responsiveness).

- During the packaging of data into messages, serialization consumes a significant amount of memory. Monitor gargage collection activities for SAP Mobile Server. If required, configure JVM heap settings to minimize garbage collection, especially during messaging application deployments and initial synchronizations.

Performance Considerations for Relay Server and Outbound Enabler

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

Table 7. Recommendations for 64-bit Production Servers

Component	Function	Default	Production Recommendation
Relay Server shared memory	Settings for shared memory buffer. Remember that shared memory must account for concurrent connections, especially with large payloads.	10MB	2GB
Relay Server response times	To decrease the small payload response time of a Relay Server that is lightly loaded, an administrator can add `up_pad_size=0` to the Relay Server configuration file created from SAP Control Center. Adding this value may have adverse affects for a highly loaded server since this setting defeats the built-in network Nagle's algorithm, which delays sending small packets in order to improve overall throughput for many small packets.	None	0

Component	Function	Default	Production Recommendation
Outbound Enabler deployment	With SQL Anywhere®, each RSOE provides an upload and download channel to the relay server.	1 Outbound Enabler per server	3 Outbound Enablers per synchronization (replication or messaging) port, especially when initial synchronizations are larger than 4MB

SAP Mobile Server Messaging Performance Tuning Reference

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

The majority of messaging tuning revolves around SAP Mobile Server configuration:

Table 8. Recommendations Generally Suitable for 64-bit Production Servers

Property	Function	Default	Production Recommendation
Number of inbound queue*	The concurrent number of threads that service package(s) requests from devices. If the processor of either the messaging database or the SAP Mobile Server is excessively high, you can throttle the number of requests and limit contention for resources using this setting. Low settings decrease parallelism; high settings may cause undue contention for resources.	5 per cluster	100 per cluster in a 2-node cluster
Number of outbound queue*	The number of outbound queues between SAP Mobile Platform and Messaging Services. Messages from these queues are eventually persisted, by the JmsBridge component into a per-device queue belonging to the Messaging Services. Subscription requests can generate a large number of outbound messages and temporarily cause backups in the queues. Since multiple devices can share the same output queue, larger number of queues can reduce delay getting the messages to the intended devices.	25 per cluster	100 per cluster in a 2-node cluster

Property	Function	Default	Production Recommendation
Check interval for package	Specifies whether the interval system should check for a package to see if there are changes to be pushed to the devices. Align this setting when the cache is being refreshed or updated. If the cache is refreshed every 4 hours, the check interval should also be 4 hours. However, if the cache is only updated via DCN, align the update frequency with the DCN interval accordingly. The check/push generation algorithm is scheduled to be enhanced in future versions.	10 minutes	10 minutes
Number of push processing queues*	Number of queues (threads) that execute the change detection function (download SQL execution) to determine if there are changes to be pushed to the device. The number of queues determines the maximum concurrency for change detection. All packages shared the same set of push processing queues.	25 per cluster	25 per cluster in a 2-node cluster.
JVM minimum heap size	The minimum memory allocated to the differencing and cache management functions of the server.	512MB	2GB
JVM maximum heap size	The maximum memory allocated to the differencing and cache management functions of the server.	2GB	6GB

* Cluster-affecting property

Messaging Maximum Content Sizes

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

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

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

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

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

Tuning Synchronization for Messaging Payloads

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

1. Isolate the monitoring database from the cache server.

   See *Isolating the Monitoring and Domain Logging Databases from Cache and Messaging Databases*.
2. Disable all enabled monitoring profiles that currently exist.

   Normally, monitoring in a cluster configuration occurs at two levels — the cluster and the domain. SAP recommends that you turn off monitoring when assessing performance.

   a) In navigation pane of SAP Control Center, click **Monitoring**.
   b) In the administration pane, select the profile name and click **Disable**.
   c) Validate that all monitoring is disabled by opening *SMP_HOME*`\Servers \UnwiredServer\Repository\Instance\com\sybase\sup\server \monitoring\ MonitoringConfiguration \domainLogging.properties`, and verifying that “status”, “enabled”, and “autoStart” are set to false.

   See *Monitoring Profiles* in *SAP Control Center for SAP Mobile Platform*.
3. Stop all SAP Mobile Servers in your cluster(s).
4. In SAP Control Center, change the number of outbound and inbound queues to 100 each on the cluster.

   a) In the left navigation pane, click **Configuration**.
   b) In the right administration pane, click the **General** tab.
   c) Select **Performance** from the menu bar.
   d) Change the default values for inbound and outbound queues to 100 each.
5. Restart all servers in your cluster(s).
6. In SAP Control Center, deploy and test a messaging package with a representative amount of data for initial subscription.

   For details, see *Deploying MBO Packages* and *MBO Package Management* in *SAP Control Center for SAP Mobile Platform*.

   For example, if the use case dictates that during an initial subscription, the mobile application is to receive 2MB of data, develop the test package to reflect that fact.

7. Start testing by using expected number of devices to perform an initial subscription, and determine if the time to get the initial data set is satisfactory for all devices.

   The maximum messaging throughput is 70 messages per second in a wired environment.
   - If the calculated throughput for the test is below this number, it is likely that the connection method (as opposed to the server environment) is the limiting factor. In this case, more devices can be supported without any degradation in server performance.
   - If the test reaches the maximum throughput, the number of devices performing the initial subscription is the maximum one server can handle. Add another server to the cluster for additional message processing power (up to a 60% increase).

See also

- *Configuring SAP Mobile Server General Properties* on page 20

LoadRunner Extension for SAP Mobile Platform

HP LoadRunner is an automated performance and testing software for examining system behavior and performance under actual load conditions. Use the SAP Mobile Platform extension to integrate with this product.

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

By default, the LoadRunner Extension is installed in *SMP_HOME*\Servers\LoadRunnerAPI. However, SAP Mobile Platform does not install LoadRunner: you must install a compatible version of LoadRunner separately, as specified in *Supported LoadRunner Versions* in *Supported Hardware and Software*. The extension is designed to be used by experienced LoadRunner testers.

The Extension has two components:

- **Code generator** - reads an XML recording file that you create using the SAP Mobile Server, and generates C# script files that LoadRunner can execute.
- **Runtime library** - provides APIs, referenced by the generated C# script files, that LoadRunner uses in addition to its own APIs.

See also

- *Performance Considerations for Relay Server and Outbound Enabler* on page 85
- *Performance Considerations for Replication* on page 87
- *Performance Considerations for Messaging* on page 96
- *Enabling RBS Performance Testing with LoadRunner* on page 109

Enabling Performance Testing with LoadRunner

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

1. *Enabling and Disabling XML Trace Recording*

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

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

   Use the extension to create C# script files for LoadRunner.
4. *Parameterizing Generated Scripts*

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

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

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

   Assign the HWC application to a connection template.

Enabling and Disabling XML Trace Recording

Control when the SAP Mobile Server records communication between the client and the server.

Prerequisites

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

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

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

Recording XML Trace Files

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

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

The XML recording files are written in *SMP_HOME*\Servers\UnwiredServer\logs\XMLRecording XML.

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

Notes:

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

Generating LoadRunner Extension Script Files

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

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

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

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

```
VuGenHwc\
VuGenOdp\
```

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

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

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

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

Parameterizing Generated Scripts

Modify the generated scripts to support access by multiple users.

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

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

In addition to the LoadRunner API documentation, see the API documentation for the SAP Mobile Platform LoadRunner extension, which is installed in *SMP_HOME*`\Servers\LoadRunnerAPI\docs\api`.

For example, consider this generated Action method:

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

```
        hwc.Value("Customer_create_fname_paramKey", hwc.STRING,  _
          key_Customer_create_fname),
        hwc.Value("Customer_create_lname_paramKey", hwc.STRING,  _
          key_Customer_create_lname),
        hwc.Value("Customer_create_address_paramKey", hwc.STRING,  _
          key_Customer_create_address),
        hwc.Value("Customer_create_city_paramKey", hwc.STRING,  _
          key_Customer_create_city),
        hwc.Value("Customer_create_state_paramKey", hwc.STRING,  _
          key_Customer_create_state),
        hwc.Value("Customer_create_zip_paramKey", hwc.STRING,  _
          key_Customer_create_zip),
        hwc.Value("Customer_create_phone_paramKey", hwc.STRING,  _
          key_Customer_create_phone),
        hwc.Value("Customer_create_company_name_paramKey",  _
          hwc.STRING, key_Customer_create_company_name),
        hwc.Value("ErrorLogs", hwc.LIST)
      )
  );
  hwc.ExpectScreen(m1, "Customercreate");
  lr.end_transaction(tx, lr.PASS);
}
catch (System.Exception _ex_1)
{
  lr.error_message(_ex_1.ToString());
  lr.end_transaction(tx, lr.FAIL);
}

return 0;
}
```

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

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

Compiling Extension Scripts and Configuring LoadRunner

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

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

   In the output folder, LoadRunner generates stub C# source files for `vuser_init.cs`, `Action.cs`, and `vuser_end.cs`. It also generates a C# project file, `Script.csproj`.
2. From your extension output folder, copy the C# script files that you generated using the extension, and paste them into the location of the empty LoadRunner-generated files that you generated in step 1.

This overwrites the empty stub files.

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

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

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

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

   a) In *SMP_HOME*`\Servers\MessagingServer\Bin\Mobile Workflow`, note the ID and version number for the redeployed Hybrid App.
   b) Compare these values to the ID and version number in the `vuser_init.cs` file. If the values are different, correct the values in `vuser_init.cs`.
5. In the LoadRunner-generated C# project, add a reference to *SMP_HOME*`\Servers\UnwiredServer\LoadRunnerAPI\lib\SAP.Mobile.LoadRunner.dll`. A convenient method is to open `Script.csproj` in Visual Studio, add the reference there, then save the project and exit Visual Studio.

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

   To work around this problem, edit the `Script.sln` file, and change the Format Version from 10.00 to 9.00. The problem does not occur if you use LoadRunner 11.5.
6. Compile the script in Virtual User Generator.
7. In Virtual User Generator, open Run-time Settings. In the Miscellaneous section under Multithreading, choose **Run Vuser as a process**.

   The default Run Vuser as a process option is not supported.
8. Copy the `DLL` files from *SMP_HOME*`\MobileSDK`*version*`\Win32` to the `bin` subfolder of your LoadRunner script directory.

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

Configuring Security

Configure your application connection template to handle simulated test users.

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

application connection template to use the test security configuration. Here are some configuration alternatives:

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

Registering HWC Application Connections

Assign the HWC application to a connection template.

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

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

No assignment is required to test ODP applications.

Enabling RBS Performance Testing with LoadRunner

Use the C# mini application method to perform load testing of RBS in SAP Mobile Platform with LoadRunner.

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

See also

- *Performance Considerations for Relay Server and Outbound Enabler* on page 85
- *Performance Considerations for Replication* on page 87

CHAPTER 11 Backup and Recovery

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

Complete backup of a host includes:

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

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

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

Backup and Recovery of the SAP Mobile Platform Data Tier

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

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

The backup and recovery life cycle has having multiple stages.

Table 9. Backup and Recovery: Stages and Activities

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

Planning Backup and Recovery

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

1. *Understanding the Installation Environment*

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

   The backup frequency of SAP Mobile Platform is determined by a variety of factors.

See also

- *Backing Up Runtime Data with a Recovery Server* on page 115
- *Maintaining Backups* on page 120
- *Diagnosing and Recovering from Failure by Scenario* on page 121
- *Performing Postrecovery Activities* on page 123

Understanding the Installation Environment

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

See also

- *Determining the Backup Frequency and Location* on page 115

SQL Anywhere Storage

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

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

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

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

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

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

Building Connections Strings for SAP Mobile Platform

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

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

1. Use this syntax to construct a connection string:

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

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

2. Test the connection:
   `dbisql -c connection-string`

   For example, you can access the cache on the data tier installed in a cluster environment with:

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

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

Determining the Backup Frequency and Location

The backup frequency of SAP Mobile Platform is determined by a variety of factors.

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

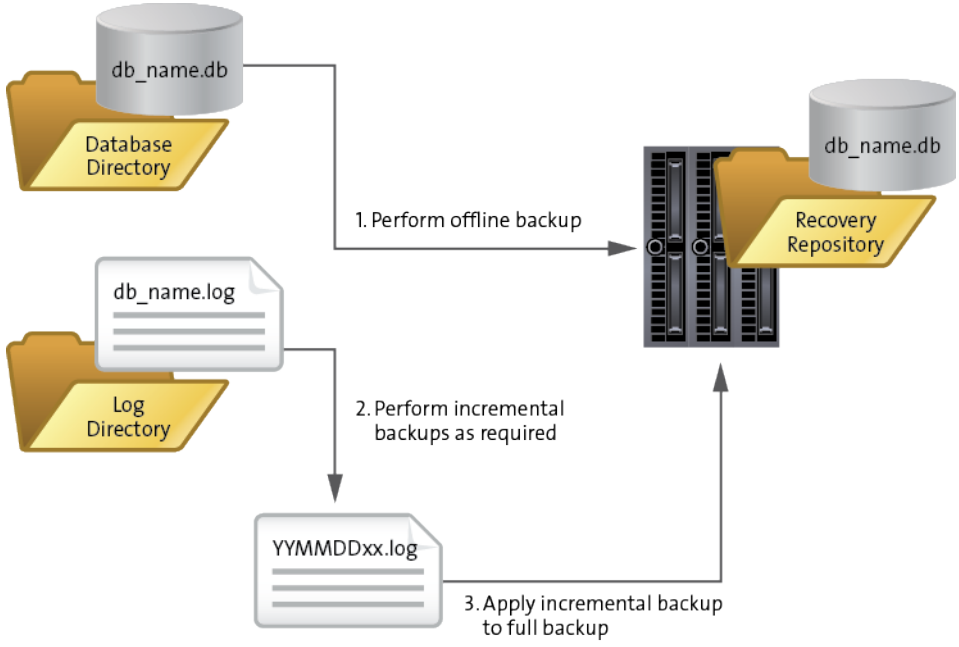
   The best practice is to backup to an entirely different physical location. For a list of database and log files you can back up, see *SQL Anywhere Storage*.

See also

- *Understanding the Installation Environment* on page 113
- *SQL Anywhere Storage* on page 113

Backing Up Runtime Data with a Recovery Server

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



1. *Enabling Transaction Log Mirroring*

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

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

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

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

   Update the full backup with contents of the transaction log.

See also

- *Planning Backup and Recovery* on page 112
- *Maintaining Backups* on page 120
- *Diagnosing and Recovering from Failure by Scenario* on page 121
- *Performing Postrecovery Activities* on page 123

Enabling Transaction Log Mirroring

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

1. Identify the target destination for transaction logs.

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

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

See also

- *Creating a Recovery Server* on page 117

Transaction Log Mirrors

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

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

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

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

Creating a Recovery Server

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

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

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

See also

- *Enabling Transaction Log Mirroring* on page 116

Performing a Full Offline Backup After Install or Upgrade

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

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

See also

- *Performing Periodic Incremental Transaction Log Backups Online* on page 118
- *SAP Mobile Platform Windows Services* on page 237

Securing Backup Artifacts

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

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

Performing Periodic Incremental Transaction Log Backups Online

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

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

1. *Using dbbackup to Create Incremental Backups*

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

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

See also

- *Performing a Full Offline Backup After Install or Upgrade* on page 117
- *Applying Increments to Full Backups on the Recovery Server* on page 119

Using dbbackup to Create Incremental Backups

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

dbbackup is in `SMP_HOME\Servers\SQLAnywhere12\BIN`*XX*.

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

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

   where:

   - **-y –** creates the backup directory or replaces a previous backup file in the directory without confirmation. If you want to be prompted, do not specify **-y**.

- **-t –** create a backup that can be used as an incremental backup since the transaction log can be applied to the most recently backed up copy of the database files.
- **-n –** changes the naming convention of the backup transaction log file to *yymmddxx*.log. *xx* are sequential letters ranging from **AA** to **ZZ**. This indexed file naming convention allows you to keep backups of multiple versions of the transaction log file in the same backup directory.
- **-x –** backs up the existing transaction log, deletes the original log, and then starts a new transaction log. Do not use this option if you are using database mirroring.

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

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

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

Using dbtran to Validate the Backup

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

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

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

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

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

Applying Increments to Full Backups on the Recovery Server

Update the full backup with contents of the transaction log.

Prerequisites

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

Task

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

You can find all utilities in the SMP_HOME\Servers\SQLAnywhere12\BIN32 folder.

1. Apply the incremental transaction log to the full backup by running:
   ```
   dbeng12 -o recovery.log db_name.db -a file\yymmddxx.log
   ```
   This command updates the full backup file (*db_name*.db) to the current state of the transaction log backup. This is the new baseline full backup from which you can recover failed environments. To update the backup again, reapply the next increment.

   Note: Database files (for example, `default.db`, or `clusterdb.db`) are stored in a parent folder, whereas transaction logs are stored in a subfolder of the same name as the database (for example, for `default.db` the transaction log is output to `..\default\120616.log`.
2. Replace the *db_name*.db in the data directory of the restore server and restart SAP Mobile Platform data services.
3. Verify the full backup by running:
   ```
   dbvalid -c connection-string
   ```

See also

- *Performing Periodic Incremental Transaction Log Backups Online* on page 118

Maintaining Backups

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

See also

- *Planning Backup and Recovery* on page 112
- *Backing Up Runtime Data with a Recovery Server* on page 115
- *Diagnosing and Recovering from Failure by Scenario* on page 121
- *Performing Postrecovery Activities* on page 123

Monitoring the Incremental Backup State

Monitoring the completion of your SAP Mobile Platform database backups is critical to success of your recovery plan.

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

1. (Only required if using log mirrors) Always check the the log mirror, to ensure there are no lags and that they are exact replicas of the source transaction log at a given moment.
2. If you are using a third-party tool for performing the incremental backup, check that the scheduled jobs are running and that the transaction logs are successfully applied to the backup database.

3. Perform routine verification of the backup by running:
   `dbvalid -c connection-string`

Performing Ad Hoc Backup and Recovery Tests

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

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

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

Diagnosing and Recovering from Failure by Scenario

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

See also

- *Planning Backup and Recovery* on page 112
- *Backing Up Runtime Data with a Recovery Server* on page 115
- *Maintaining Backups* on page 120
- *Performing Postrecovery Activities* on page 123

Database Failure

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

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

Recovery required

1. Stop all SAP Mobile Platform services on all SAP Mobile Server and data tier nodes in the cluster.
2. Apply the transaction log to the backup database.
3. Replace the primary database with the backup database.

Hardware Failure: Mirrors Available

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

Recovery required:

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

Hardware Failure: No Mirrors Available

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

Recovery required:

1. Stop all SAP Mobile Platform services.
2. Replace the primary database with the backup database.

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

Backup Database Is Lost

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

Symptoms: No back up is available.

Recovery required:

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

Database Corruption

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

Verifcation required:

To verify corruption, run:

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

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

Recovering from Server Node Software Corruption or Host Failure

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

Recovery required:

1. Ensure SAP Mobile Server services on the data tier node are running.
2. If you are recovering a software component with the installer:
   a) Uninstall the affected component.
   b) Reinstall the affected component.
   c) If the component is a data tier component, copy the database file from the recovery server.
3. If you are recovering a software component with a clone or snapshot image, revert to the most recent backup.

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

See also

- *Applying Increments to Full Backups on the Recovery Server* on page 119

Performing Postrecovery Activities

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

See also

- *Planning Backup and Recovery* on page 112
- *Backing Up Runtime Data with a Recovery Server* on page 115
- *Maintaining Backups* on page 120
- *Diagnosing and Recovering from Failure by Scenario* on page 121

Synchronizing Data from Device Applications

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

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

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

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

2. Resolve this issue by asking the device user to:
   a) From the application, initiate a database reset.

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

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

See also

- *Resubmitting Lost Transactions* on page 124
- *Reregistering Applications* on page 124

Resubmitting Lost Transactions

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

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

   Note: Other symptoms that are based on the actual application design may exist; work with the development team to evaluate those application specific symptoms that may arise.

2. Resolve the lost transaction issue by requesting that the user:
   a) Reregister the application connection.
   b) Resubmit all transactions from the application after it is reregistered.

See also

- *Synchronizing Data from Device Applications* on page 123
- *Reregistering Applications* on page 124

Reregistering Applications

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

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

This message shows that an unregistered device has tried to connect. No message is sent or received between this device and SAP Mobile Server.

2. Resolve the issue by asking the device user to reregister the application connection.

See also

- *Synchronizing Data from Device Applications* on page 123
- *Resubmitting Lost Transactions* on page 124

Backup and Recovery of Security Artifacts

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

Backing Up and Recovering Messaging Keys

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

Registry entries for the messaging keys are recorded in `Computer\HKEY_LOCAL_MACHINE\Software\Wow6432Node\Sybase\SybaseMessagingServer\Server\identinfo`. If you are using an alternate method for creating complete backups of a host system, continue to use that method to backup and recover these keys.

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

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

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

   Note: If this key becomes compromised, the security of messaging communications can no longer be guaranteed.
3. Delete the `C:\Windows\System32\msgserver.reg` file.
4. To recover a system that has failed:
   a) Reinstall SAP Mobile Platform.
   b) Stop SAP Mobile Server.

c) Copy the `msgserver.reg` file from its secure location to a temporary location on the server that is being restored (for example `C:\temp`).

d) From a command prompt, run:

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

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

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

e) Delete the `msgserver.reg` file from it's temporary location.

f) Restart SAP Mobile Server.

Backing Up and Recovering Other System Keys and Certificates

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

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

CHAPTER 12 Platform Monitoring and Diagnostics

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

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

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

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

System Monitoring Overview

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

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

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

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

- Cache activity

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

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

Table 10. System monitoring tasks

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

Monitor

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

Monitored operations include security, replication-based synchronization, messaging-based synchronization, messaging queue, data change notification, device notification, package,

user, and cache activity. These aspects of monitoring are important to ensuring that the required data is collected.

The critical aspects of monitoring include:

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

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

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

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

Monitoring Profiles

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

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

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

Planning for System Monitoring

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

1. Understand the business requirements your system is addressing.

2. Translate those business needs into performance objectives. As business needs evolve, performance objectives must also evolve.
3. Perform capacity planning and evaluate performance of the monitoring database.

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

Next

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

Creating and Enabling a Monitoring Profile

Specify a monitoring schedule for a group of packages.

Prerequisites

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

Task

1. Open and log in to SAP Control Center.
2. In the left navigation pane, select **Monitoring**.
3. In the right administration pane, select the **General** tab.
4. Click **New** to create a monitoring profile.
5. Enter a name for the new profile.
6. Select the **Domains and Packages** tab and choose packages to be monitored according to these options:
   - Monitor all domains and packages – select **All Domains and Packages**.
   - Monitor all packages from one or more domains – select a domain, then click **Select All Packages**. Perform this step for each domain you want to monitor.

- Monitor specific packages from one or more domains – select a domain, then select the particular packages you want to monitor from that domain. Perform this step for each domain you want to monitor.

7. Select **View my selections** to view the packages you selected for the monitoring profile. Unselect this option to return to the package selection table.
8. Select **Enable after creation** to enable monitoring for the selected packages immediately after you create the profile. By default, this option is selected. Unselect this option to enable the monitoring profile later.
9. On the **Schedule** tab, select a schedule to specify when monitoring takes place:
   - **Always On** – this schedule requires no settings. Package activity is continually monitored.
   - **Run Once** – specify a length of time during which monitoring occurs, in either minutes or hours. Package activity is monitored for the duration specified for one time only.
   - **Custom** – specify start and end dates, start and end times, and days of the week. Package activity is monitored according to the time frame specified. See *Setting a Custom Monitoring Schedule*.
10. Click **OK**.
    A status message appears in the administration pane indicating the success or failure of profile creation. If successful, the profile appears in the monitoring profiles table.
11. To enable a profile that you did not enable during creation, select the monitoring profile and click **Enable**.

Setting a Custom Monitoring Schedule

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

Prerequisites

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

Task

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

- Select **End Date** to set a date that ends the monitoring of package activity. To be more specific, you can also enter an **End Time**.
- Select the days of the week that package monitoring runs. This means that for the days you select, the schedule runs every week on the day or days you specify.

If you do not indicate a time frame, SAP Mobile Server uses the default custom schedule, which is equivalent to Always On monitoring.

4. Click **OK**.

Configuring Monitoring Performance Properties

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

Prerequisites

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

Task

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

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

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

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

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

specified time interval. Enter the desired rows and duration adjacent to **Rows** and **Minutes**, respectively.

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

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

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

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

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

8. Click **OK**.

See also

- *Monitoring Database Schema* on page 366

Monitoring Usage

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

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

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

Also note that the processing times are calculated based on the time the request (or message) arrives on the server, and the time it took to process the request (or message) on the server. The

client-side time (request origin time, and time taken to deliver to the server) are not factored into that data.

Reviewing System Monitoring Data

Review data for monitored activities in SAP 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 SAP Control Center.
2. In the left navigation pane, select **Monitoring**.
3. In the right administration pane, select one of the following tabs according to the type of monitoring data you want to view:

   - **Security Log**
   - **Replication**
   - **Messaging**
   - **Queue**
   - **Data Change Notifications**
   - **Device Notifications**
   - **Package Statistics**
   - **User Statistics**
   - **Cache Statistics**

Current and Historical Data

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

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

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

Performance Data: KPIs

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

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

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

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

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

Performance Statistics

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

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

Use the Monitoring node to:

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

Monitoring Data Categories

Monitoring data is organized according to object type, allowing administrators to perform focused data analysis on specific activities and SAP Mobile Platform components. Current,

historical, and performance-based statistics facilitate user support, troubleshooting, and performance tracking for individual application environments.

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

Security Statistics

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

Security Log Statistics

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

User security data falls into these categories:

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

Replication Statistics

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

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

Current Replication Statistics

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

SAP Mobile Server monitors replication requests using these statistical categories:

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

Replication History Statistics

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

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

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

Table 11. Detail view information

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

Table 12. Summary view information

Category	Description
Application ID	The ID number associated with an application.
User	The name of the user associated with the device ID.
Package	The package name.
Total Rows Sent	The total number of rows sent during package synchronization.

Category	Description
Total Operation Replays	The total number of operation replays performed by clients during synchronization.
Total Bytes Sent	The total amount of data (in bytes) downloaded by clients from SAP Mobile Server during synchronization.
Total Bytes Received	The total amount of data (in bytes) uploaded to SAP Mobile Server by clients during synchronization.
Start Time	The date and time that the synchronization request was initiated.
Total Synchronization Time	The amount of time taken to complete the synchronization.
Total Errors	The total number of errors that occurred for the package during synchronization.
Domain	The domain to which the package involved in synchronization belongs.

Replication Performance Statistics

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

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

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

KPI	Description
Total Distinct Package Synchronized	The total number of packages subject to synchronization.
Total Distinct Users	The total number of users who initiated synchronization requests. This value comprises only individual users, and does not count multiple synchronizations requested by the same user.
Average/Minimum/Maximum Synchronization Time	The average, minimum, or maximum amount of time SAP Mobile Server took to finish a complete synchronization.
Time at Minimum/Maximum Synchronization Time	The time of day at which the shortest or longest synchronization completed.
Package with Minimum/Maximum Synchronization Time	The name of the package and associated MBO with the shortest or longest synchronization time.

KPI	Description
Average/Minimum/Maximum MBO Rows Per Synchronization	The average, minimum, or maximum number of MBO rows of data that are downloaded when synchronization completes.
Average/Minimum/Maximum Operation Replays per Synchronization (records received)	The average, least, or greatest number of operation replays per synchronization received by SAP Mobile Server from a client.
Total Bytes Sent	The total number of bytes downloaded by clients from SAP Mobile Server.
Total Bytes Received	The total number of bytes uploaded from clients to SAP Mobile Server.
Total Operation Replays	The total number of operation replays performed on the EIS.
Total Errors	The total number of errors that took place across all synchronizations.
Average/Minimum/Maximum Concurrent Users	The average, least, or greatest number of users involved in concurrent synchronizations.
Time at Minimum/Maximum Concurrent Users	The time at which the least or greatest number of users were involved in concurrent synchronizations.

Messaging Statistics

Messaging statistics report on messaging synchronization activity for monitored packages.

- Current monitoring data tracks the progress of messages from device users presently performing operation replays or synchronizing MBOs.
- Historical data reveals statistics indicating the efficiency of completed transactions.
- Performance monitoring provides an overall view of messaging payload activity intended to highlight areas of strength and weakness in the application environment.

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

Current Messaging Statistics

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

SAP Mobile Server monitors messaging requests using these categories:

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

Messaging History Statistics

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

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

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

Table 13. Detail view information

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

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

Table 14. Summary view information

Category	Description
Application ID	The ID number associated with an application.
User	The name of the user associated with the device ID
Package	The package name
Total Messages Sent	The total number of messages sent by SAP Mobile Server to clients during synchronization
Total Messages Received	The total number of messages received by SAP Mobile Server from clients during synchronization
Total Payload Size Sent	The total amount of data (in bytes) downloaded by clients from SAP Mobile Server during synchronization
Total Payload Size Received	The total amount of data (in bytes) uploaded to SAP Mobile Server by clients during synchronization

Category	Description
Total Operation Replays	The total number of operation replays performed by clients during synchronization
Last Time In	The date and time that the last inbound request was received
Last Time Out	The date and time that the last outbound response was sent
Subscription Commands Count	The total number of subscription commands sent during synchronization; for example, subscribe, recover, suspend, and so on
Total Errors	The number of errors that occurred for the package during synchronization
Domain	The domain to which the package involved in synchronization belongs

Messaging Performance Statistics

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

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

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

KPI	Description
Total Messages	The total number of messages sent between the server and clients during synchronization.
Total Distinct Devices	The total number of devices involved in synchronization. This total includes the same user multiple times if he or she has multiple devices. The value comprises individual devices, and does not count multiple synchronizations requested by the same device.
Total Distinct Users	The total number of users who initiated synchronization requests. This value comprises individual users, and does not count multiple synchronizations requested by the same user if he or she uses multiple devices.
Average/Minimum/Maximum Concurrent Users	The average, minimum, or maximum number of users involved in simultaneous synchronizations.
Time at Minimum/Maximum Concurrent Users	The time at which the greatest or least number of users were involved in concurrent synchronizations.

KPI	Description
Average/Minimum/Maximum Processing Time	The average, minimum, or maximum amount of time SAP Mobile Server took to respond to a sync request message.
Time at Minimum/Maximum Message Processing Time	The time of day at which the shortest or longest message processing event completed.
MBO for Maximum/Minimum Message Processing Time	The name of the package and associated mobile business object (MBO) with the shortest or longest message processing time.
Average/Minimum/Maximum Message Size	The average, smallest, or largest message sent during synchronization.
Total Inbound Messages	The total number of messages sent from clients to SAP Mobile Server.
Total Outbound Messages	The total number of messages sent from SAP Mobile Server to clients.
Total Operation Replays	The total number of operation replays performed by clients on MBOs.
Total Errors	The total number of errors that took place across all synchronizations.

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

Messaging Queue Statistics

Messaging queue statistics reflect the status of various messaging queues. The data does not reveal any application-specific information, but provides a historical view of messaging activities that communicates the efficiency of messaging-based synchronization, as well as the demands of device client users on the system.

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

Messaging Queue Status

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

These key indicators monitor messaging queue status:

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

Data Change Notification Statistics

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

Monitoring DCN statistics allows you to troubleshoot and diagnose performance issues if, for example, the cache is not being updated quickly enough. These statistics help to identify which packages took longest to process data changes, as well as times of peak performance or strain on the system.

Data Change Notification History Statistics

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

Details about past notification events are organized into these categories:

Category	Description
Domain	The domain to which the package affected by the DCN belongs.
Package	The name of the package containing data changes.

Category	Description
MBO	The name of the MBO to which the notification applied.
Notification Time	The date and time that SAP Mobile Server received the DCN.
Processing Time	The time that SAP Mobile Server used to process the DCN.

Data Change Notification Performance Statistics

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

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

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

Key performance indicator	Description
Total Notifications	The total number of notifications sent by the enterprise information system to SAP Mobile Server.
Average/Minimum/Maximum Processing Time	The average, minimum, or maximum amount of time SAP Mobile Server took to process a DCN.
Time at Minimum/Maximum Message Processing Time	The time of day at which the shortest or longest DCN processing event completed.
Time of Last Notification Received	The time at which the most recent DCN was received by SAP Mobile Server.
MBO with Minimum/Maximum Notification Processing Time	The name of the package and associated mobile business object (MBO) with the shortest or longest notification processing time.

Device Notification Statistics

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

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

Device Notification History Statistics

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

Details about past device notification events fall into these categories:

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

Device Notification Performance Statistics

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

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

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

KPI	Description
Synchronization Group for Maximum Notifications	The synchronization group for which the maximum number of notifications were sent.
Package for Maximum Notifications	The package for which the greatest number of device notifications were sent.
Total Notifications	The total number of device notifications sent from SAP Mobile Server to devices.

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

Package Statistics

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

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

Replication Package Statistics

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

These key indicators monitor replication packages:

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

- Total Bytes Received
- Total Bytes Sent
- Total Operation Replays
- Total Devices
- Total Rows Sent
- Total Rows Received

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

Messaging Package Statistics

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

Note: The Total Subscription Commands KPI is not applicable at the MBO level.

These key indicators monitor messaging packages:

KPI	Description
Total Subscription Commands	The total number of subscription commands sent from clients to the server.

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

User Statistics

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

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

Note: These statistics are not supported for SAP Mobile CRM and SAP Hybrid App for SAP application users.

Replication User Statistics

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

These key indicators monitor replication users:

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

Messaging User Statistics

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

These key indicators monitor messaging users:

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

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

Cache Statistics

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

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

Note: These statistics are not supported for SAP Mobile CRM and SAP Hybrid App for SAP application users.

MBO Statistics

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

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

Key performance indicator	Description
Cache Group	The name of the group of MBOs associated with this cache activity.
MBO	The name of the single mobile business object associated with this cache activity.

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

Cache Group Status Statistics

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

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

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

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

Refining Scope with Filters, Sorting, and Views

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

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

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

Exporting Monitoring Data

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

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

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

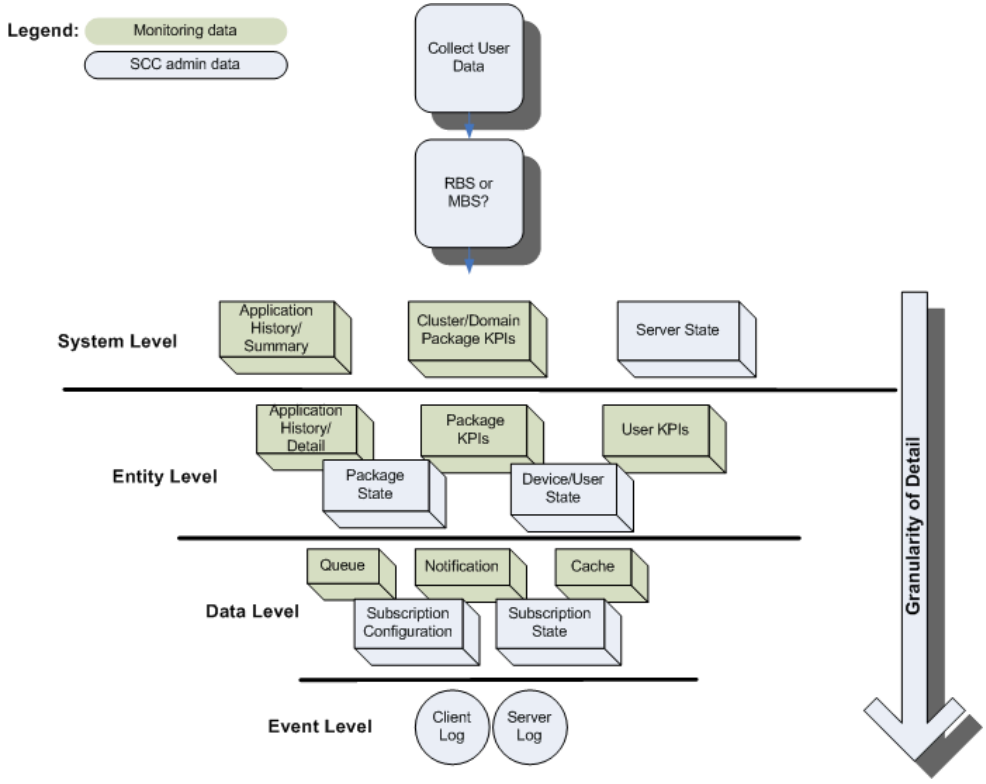
1. In the left navigation pane, select **Monitoring**.
2. In the right administration pane, select the tab corresponding to the monitoring data you want to view.
3. Perform a search using the appropriate criteria to obtain the desired monitoring data.
4. Click **Export**.
5. Select a file type for the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

Monitoring Data Analysis

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

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

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



See also

- *Device Application Performance or Issue Analysis* on page 157
- *Access Denied Analysis* on page 161
- *Data Update Failure Analysis* on page 162

Collecting Data

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

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

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

Device Application Performance or Issue Analysis

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

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

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

See also

- *Collecting Data* on page 156
- *Checking Package/User Histories* on page 157
- *Checking Overall Package Statistics* on page 158
- *Checking User KPIs for Other Data Usage* on page 159
- *Checking Server Responsiveness in SAP Control Center* on page 160
- *Looking for Errors and Failures in SAP Control Center* on page 161

Checking Package/User Histories

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

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

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

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

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

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

Next

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

Checking Overall Package Statistics

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

1. In SAP Control Center, click the Monitoring node, then click the **Package Statistics** tab.

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

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

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

Next

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

Checking User KPIs for Other Data Usage

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

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

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

   A high data push value might indicate that at some point, data demands were large. If the average synchronization and minimum synchronization times fall within normal ranges but the maximum is high, it may indicate that data volumes have created a synchronization performance lag for the package. Low data delivery for the package (determined during package-level analysis), but high values for the individual user may suggest an unusual demand pattern which may be harmless or require further investigation. Simultaneous

synchronization demands may be impacting the performance of all applications on the device.

Next

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

Checking Server Responsiveness in SAP Control Center

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

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

Next

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

Reviewing the Pending Items Count for Messaging Applications

Use the count in the Pending Items column to determine the number of undelivered messages for offline applications paired to the corresponding connection ID. When a the application moves into an online and connected state, the messages are routed to the application as soon as possible.

This column is used for Hybrid Apps and OData SDK Applications only. The count displayed in the Pending Items column is partly determined by using the number of encrypted messages saved in the QUEUED_MESSAGES table in the cache. Before delivery the message may be split into multiple "items". The split depends on how much data you send, how it is structured, and how big the message size has been configured.

1. Click **Application** > **Application Connections**.
2. Locate the connection ID in an find the corresponding **Pending Items** count for it.
   - If the value is 0, then messages have been delivered.

- Otherwise, the value indicates that a delivery is still pending. If you see a much larger value in the column count, then it could indicate that messages are large and could affect performance to some degree.

Looking for Errors and Failures in SAP Control Center

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

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

1. In SAP Control Center, click the **Packages** node.
2. To check whether the problem exists in either the device or the EIS data source:
   a) Click the **Client Log** tab.
   b) Check the Operation and Message columns and look for any operation replays that failed with error code 500, and the reason for the failure. For example, the client log shows all logs on device. For failed operations on the device, check the details of error message and assess what kind of error may have occurred.
3. To check whether the problem exists in either the server or the EIS data source:
   a) Expand the package tree until MBOs and operations for the package complete the tree navigation for the package.
   b) For each MBO and operation, select the node in the navigation tree, then click the **History** tab.
   c) Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the duration you require.
   d) Review the data and look for any errors and the potential causes of those errors. You can check the error history, the exception or error about the operation is listed in the details of that error. Use the message to reveal the issue and coordinate with the development team as required.

Access Denied Analysis

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

Checking the Security Log

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

1. In SAP 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 SAP Control Center, expand the navigation tree in the SAP Mobile 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 SAP Mobile Platform user is the exact name of the user in the security repository, then no mapping is required.
     - If the SAP Mobile Platform user differs, even slightly, then logical roles used by the package, and physical roles used in the repository must be manually mapped.
   - Review the existing security policy with the security administrator to ensure that privileges are set correctly.

Data Update Failure Analysis

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

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

Checking Last Notification Timestamp

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

Prerequisites

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

Task

1. In SAP Control Center, click the Monitoring node, then click the **Device Notifications** tab.
2. Select **History**.
3. Click **User** to sort on user name in ascending or descending alphabetical order.
4. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the time frame you specify.
5. In the Time of Notification column, ensure that the timestamp was recently recorded, then compare the start and end time of the last known synchronization. Device notification should always occur before a synchronization. If a notification is not received by a user, they may assume that data is not updating correctly.

Checking Cache Statistics

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

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

not working as designed. See *Validating Settings of Features that Update Data in SAP Control Center* to see how the subscription that schedules push notifications is set up.

Validating Settings of Features that Update Data in SAP Control Center

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

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

1. Check the settings of the cache group used by the package.
   a) In SAP Control Center, expand the Packages node, select the package name, then choose the **Cache Group** tab.
   b) Select the box beside the cache group name and click **Properties**.
   c) Verify:
      - The cache interval or schedule used to refresh data in the SAP Mobile Server cache based on changes in the back-end EIS data source. Make note of that refresh interval to use in the next step.
2. Select the **Subscriptions** tab and verify:
   - That the user subscription has not been removed or suspended.
   - That push synchronization is used as required. Follow up with the user to ensure that push synchronization is enabled on the device.
   - That the synchronization group change detection interval is configured appropriately based on the cache interval or schedule repeat. This value determines how frequently change detection occurs. If you see a value, of 0, then this action is disabled. Enable this action by setting an appropriate value.
   - That the notification threshold is configured appropriately based on the synchronization group interval. This value determines how frequently a user is notified of updated server cache data.

System Logs

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

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

Messages in various platform logs can provide information about:

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

- System events and failures

Log File Locations

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

Table 15. Log file locations

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

Log type	Location
Relay server and RSOE	Default log details: `%temp%\ias_relay_server_host.log` *`%temp%`* is the Windows environment variable, for example, `C:\WINDOWS\Temp`. **Note:** In the case of IIS Web servers, the log file is `C:\WINDOWS\system32\LogFiles`. However, this location can be configurable in IIS Manager. RSOE log files, by default: *`SMP_HOME`*`\Servers\UnwiredServer\logs\<`*`nodeName`*`>.RSOE<`*`n`*`>.log` For example: `RBShost1.RSOE4.log`.

Installer	`SMP_HOME`
	`SMP_HOME\InstallLogs`
	`SMP_HOME\InstallLogs\silentInstall`

Support Package Installation Log	`SMP_HOME\sdk`*`XX`*`sp`*`XX`*`[pl`*`XX`*`]logs` for the SDK and `SMP_HOME\smp`*`XX`*`sp`*`XX`*`[pl`*`XX`*`]logs`
Messaging Server Upgrade Log	*`SMP_HOME`*`\Servers\MessagingServer\Bin\scripts\DBUpgrader-Trace_YYYYMMDDHH24MMSS.txt`
Domain	In the domainlog database; viewable in SAP Control Center.
Cache database logs	By default, cache database errors are logged in the `errorlog.txt` file, located in *`SMP_HOME`*`\Servers\UnwiredServer\logs`. For a data tier installation, there are three database server logs, by default located in *`SMP_HOME`*`\Data\CDB` (but may vary depending on the installation): • `errorlog.txt` – cache DB server log • `clusterdb_errorlog.txt` – cluster DB server log • `monitordb_errorlog.txt` – monitor and domainlog DB server log

Log type	Location
SAP Mobile WorkSpace log	In SAP Mobile WorkSpace, use the **Windows** >**Show View** > **Other** > **Error Log** view, or collect the log file from the Eclipse workspace directory at `<workspace folder>/.metadata/.log file`.

Message Syntax

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

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

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

Severity Levels and Descriptions

SAP Mobile Platform can record messages at various severity levels.

You can control the level of messages that get recorded for SAP Mobile Server from SAP Control Center. See *Configuring SAP Mobile Server Log Settings* in *SAP Control Center for SAP Mobile Platform*.

Log messages that typically get recorded include:

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

Server Log

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

These server logs are available:

- SAP Mobile Server logs – collect data on SAP Mobile Server health and performance by component, and retrieve data for all or specific searches. You can save and archive system logs.
- Messaging Server logs – retrieve trace data for all or specific messages. Export data for archive or further analysis.

SAP Mobile Server Runtime Logging

SAP Mobile Server logs collect runtime information from various embedded runtime components.

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

You can view logs from:

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

Configuring SAP Mobile Server Log Settings

SAP Mobile Server logs collect data on SAP Mobile Server health and performance by component. Configure SAP Mobile Server log properties to specify the amount of detail that is written to the log, as well as the duration of the server log life cycle.

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

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

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

Alternatively, select **No limit** to retain all log files.

5. Set the HTTP log settings.
   a) Select **Enable HTTP request log** to generate an HTTP request log in the logs subdirectory. The generated log file name is *server-name*-http.log.

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

   b) Set the **Maximum file size** of the log file.
   c) If you want to back up the log file when it reaches the limit, select **Perform rotation**. The backup file is saved as *server-name*-http.log.*backup-number*.
   d) If you want to continue to use the current log file when the server restarts, select **Reuse**.If you do not select this option, when the server restarts the current log file is copied to the .\old subdirectory and a new log file is created.
   e) If you want to archive the log file select **Archive**, then specify the **Archive file name**. If you want to compress the archive log file select **Compress**.
6. For each component, choose a log level:

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

Log level	Messages logged
All	Complete system information
Trace	Finer-grained informational events than debug
Debug	Very fine-grained system information, warnings, and all errors
Info	General system information, warnings, and all errors

Log level	Messages logged
Warn	Warnings and all errors
Error	Errors only
Console	Messages that appear in the administration console only (when SAP Mobile Server is running in non-service mode)
Off	Do not record any messages

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

Log life cycle default example

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

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

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

See also

- *Configuring SAP Control Center Logging for Performance Diagnostics* on page 174
- *Configuring RSOE Logging* on page 173
- *Enabling Custom Log4j Logging* on page 176

Messaging Server Runtime Logging

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

Configuring Messaging Server Log Settings

Messaging Server logs create trace configurations for messaging modules, and retrieve trace data for all or specific messages. Configure trace configuration properties for modules to

specify the amount of detail that is written to the log. You can configure trace settings for the primary server cluster in SAP Control Center for each module. The settings are available to cluster servers through the shared data folder.

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

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

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

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

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

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

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

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

Enabling and Disabling HTTP Request Logging for DCNs

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

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

1. Open *SMP_HOME*\Servers\UnwiredServer\Repository\Instance\com\sybase\djc\server\ApplicationServer\${yourserver}.properties.
2. Delete the `enableHttpRequestLog` line.
3. Save the file.
4. Restart SAP Mobile Server.

Increasing the Maximum Post Size

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

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

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

Configuring RSOE Logging

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

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

See also

- *Configuring SAP Control Center Logging for Performance Diagnostics* on page 174
- *Configuring SAP Mobile Server Log Settings* on page 168
- *Enabling Custom Log4j Logging* on page 176

Configuring and Enabling Relay Server Logging

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

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

1. Open `rs.config`.

   The location of this file varies depending on the type of Web server used to host Relay Server.
2. Locate the `[options]` section.
3. Set these properties:

   - `start=no`
   - `verbosity=0`

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

   rshost -f rs.config -u -q -qc
5. Check the Relay Server desktop tray icon to ensure there are no errors.

   - If there are no errors, close the command window.
   - If there are errors, check the `rs.config` file for errors and try again.
6. Validate the setup by opening the log file and confirming that the log entries reflect the log level you configure.

Configuring SAP Control Center Logging for Performance Diagnostics

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

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

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

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

1. Open *SCC_HOME*\plugins\com.sybase.supadminplugin_<*SAPMobilePlatformVersion*>\agent-plugin.xml.
2. Add the following line to the file under the <properties> element:

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

3. Open *SCC_HOME*\conf\log4j.properties.
4. If you are experiencing log truncation issues, edit the following lines to change the default values for maximum file size (default: 25MB) and maximum backup index (default: 20 files) to the values shown in this example:

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

5. Restart SAP Control Center.

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

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

1. Modify the *SCC_HOME*\plugins\com.sybase.supadminplugin\agent-plugin.xml file as indicated in step 2, above.
2. Restart SAP Control Center.
3. Log in and perform your regular administrative tasks.
4. View the execution time indicators for these operations in the cookie file supatcookie.sol. The location of this file varies depending on your operating system:

Operating System	Location
Windows XP	C:\Documents and Settings\<*username*>\Application Data\Macromedia\Flash Player\#SharedObjects
Windows Vista Windows 7	C:\Users\<*username*>\AppData\Roaming\Macromedia\Flash Player\#SharedObjects

Operating System	Location
Macintosh OS X	`/Users/<username>/Library/Preferences/Macromedia/Flash Player/#SharedObjects`
Linux	`/home/<username>/.macromedia/Flash_Player/#SharedObjects`

5. Analyze the log using your preferred method of data analysis.

See also

- *Configuring SAP Mobile Server Log Settings* on page 168
- *Configuring RSOE Logging* on page 173
- *Enabling Custom Log4j Logging* on page 176

Enabling Custom Log4j Logging

Use any editor (text, XML, or an IDE) to create a `log4j.xml` file.

Prerequisites

Understand the use restrictions for log4j logging.

Note: The file format of this file falls outside the scope of this document. For details about `log4j.xml` format, see *http://logging.apache.org/log4j/*.

Task

1. In an editor, create a `log4j.xml` file.
2. Define the appenders you require and save the file. Appenders must be named, have a class defined, and require one or more parameters.
3. Add the file to all servers in your cluster. The file must be saved in: *SMP_HOME*`\Servers\UnwiredServer\Repository\`.
4. Repeat these steps on each server in the cluster.
5. Restart all servers.

See also

- *Configuring SAP Control Center Logging for Performance Diagnostics* on page 174
- *Configuring SAP Mobile Server Log Settings* on page 168
- *Configuring RSOE Logging* on page 173

Log4j Restrictions

The default SAP Mobile Server runtime logging and log4j logging are peer systems; the implementation of one does not usually impact the other, unless functionality overlaps.

Border conditions present in the `log4j.xml` configuration file may interfere with the configuration defined by *`SMP_HOME`*`\Servers\UnwiredServer\Repository\logging-configuration.xml`. Do not create appenders that output to:

- Any console (either System.out or System.err) with any appender, including ConsoleAppender. Log data destined to these appenders is ignored by SAP Mobile Platform runtime components.
- The SAP Mobile Server log. By default you can find the log file is created as *`SMP_HOME`*`\Servers\UnwiredServer\logs\<hostname>-server.log`.

Any changes you make to a `log4j.xml` configuration are applied only to the node on which the change is made; they are not propagated across the entire cluster. Therefore, you must edit the file on each node individually, as required.

Windows Event Log

SAP Mobile Platform system messages are logged in the Microsoft Windows application event log.

Events generated by respective platform components are logged with the following source identifier values.

- SAP Messaging Server
- SQLANY 12.0 (32 bit) or SQLANY64 12.0 (64-bit)
- SAP Mobile Server
- SAP Control Center *X.X*

The following events are logged.:

- Server start and stop events are recorded as information events.
- Server errors such as license errors, failure to connect to runtime databases, and so forth, are logged as error events.
- Whenever a user account is locked due to repeated login failures, a message is logged as warning event.

Domain Logs

The domain log enables an administrator to monitor application activities throughout the system. Detailed views of application activities are available by subsystem. The administrator can review activities in a specific subsystem log view, view correlated data in multiple subsystems, or view a unified log across all subsystems. The administrator must enable logging, and then use log filters to view data of interest.

By default, only error messages are recorded in each domain's log. To enable domain logging, you must create a log profile. See *Creating and Enabling Domain Logging* in *SAP Control Center for SAP Mobile Platform*.

Supported Log Subsystems

Log subsystems provide categories that enable you to filter and correlate application data at a more granular level. Understanding these subsystems enables you to formulate more specific filters for tracking application activities throughout the system.

Subsystem	Description
All	Provides a unified view of all subsystems, enabling you to look for activities, trends, and symptoms across multiple subsystems.
Synchronization	Provides a view of synchronization activities. Within this subsystem, additional categories include data synchronization, operation replay, subscription, result checker, cache refresh, and data services (DS) interface.
Device Notification	Provides a view of device notification activities.
DCN	Provides a view of data change notification (DCN) activities. Within this subsystem, additional categories include general DCN, and Hybrid App DCN.
Security	Provides a view of security-related activities.
Error	Provides a view of errors.
Connection	Provides a view of connection activities. Within this subsystem, additional categories include DOE, JDBC, REST, SAP®, and SOAP connections.
Push	Provides a few of push notification activities.
Proxy	Provides a view of Online Data Proxy connection-related activities.
Server	Provides a view of server-related activities.
Dispatcher	Provides a view of dispatcher activities. Within this subsystem, categories include Replication, Messaging, and Service.
Application	Provides a view of application activities. Within this subsystem, categories include Registration, Setting, Agentry Event and Agentry Message.

.

Managing Domain Logs

Configure settings to perform logging for your applications whenever needed.

Messages are logged for synchronization, data change notification, device notification, package, user, and cache activities among others. The following aspects of logging are

important to ensure that performance of the applications is minimally impacted and the required data is collected. In production system, it may be used to diagnose application issues. A developer may also use it in a development or test environment for debugging purposes.

The critical steps for configuring the domain log include:

1. Set up the proper domain log configuration. A domain log configuration sets the server behavior for writing data to database, automatic purge, and data source where the log data is stored. A default configuration is created for you, however you will likely want to customize this configuration for your environment.

   By default, log data is flushed every 5 minutes. In development and debugging scenarios, you may need to set the flush behavior to be immediate. Set the Number of rows and Batch size properties to a low number. You can also disable flush, which results in immediately persisting changes to the database. If you are setting up immediate persistence in a production environment, you may experience degraded performance. Use persistence with caution. This configuration can be done by platform administrator only.
2. Create a logging profile. A logging profile allows a platform administrator or domain administrator to define the target of logging. SAP Mobile Server can be configured to log messages for a specific application or package, security configuration or user, device connection, and/or backend connection type or specific one, or a combination thereof.
3. Review the captured data. A platform or domain administrator can review logged data using log filters. The filters enable you to retrieve data logged for a specific thread, application, user, connection, etc. among other options.

All of the above steps are performed in SAP Control Center. Refer to the topic group *Domain Logs* in *SAP Control Center for SAP Mobile Platform* for details.

Planning for Domain Logging

Domain logging is an option for obtaining detailed information on all aspects of device, user, application, domain, and data synchronization related activities. The capability can have an adverse influence on system performance, so planning is important before using it on a production system.

Follow these guidelines for implementing domain logging:

1. Understand the business requirements that your logging needs to address. Typical usage of the domain logging data would be for debugging application issues whenever they are reported.
2. Translate those business needs into logging objectives (how frequently it may be used, the duration of logging, amount of generated data, life span of generated log, and so forth). As business needs evolve, the logging configuration must also evolve.
3. To isolate the performance impact and meet your capacity planning outcome, you may run the monitor and domain log database on a separate host (from Cache DB and Cluster DB host).

See *Isolating the Monitoring and Domain Logging Databases from Cache and Messaging Databases*.

4. Identify the target of logging for tracking requests through the system. The logging system offers flexibility to enable logging at multiple levels of granularity, and therefore it is of utmost importance to enable logging at proper granularity level so that the necessary data is collected without incurring performance or resulting in major log data volume. Logging activities for the target data result in a major load on the system.
5. Perform some tests with simulated application loads, to evaluate performance of the domain logging database.

See also

- *Isolating the Monitoring and Domain Logging Databases from Cache and Messaging Databases* on page 66

Enabling and Configuring Domain Logging

Configure auto purge, flush threshold, and flush batch size settings to determine how long domain log data is retained, how frequently it is written to database from server nodes, and set a domain log database connection to configure where domain log data is stored.

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

1. In the left navigation pane of SAP Control Center, select **Domains**.
2. Under the default domain node, select **Log**.
3. In the right administration pane, select the **Settings** tab. These settings are used for all domains.
4. Click **Configuration**.
5. Configure auto purge settings.

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

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

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

   a) Select **Enable flush threshold** to activate flush threshold functionality.

Note: Enabling flush configuration is a good practice for performance considerations. Be aware there may be a consequent delay in viewing data, until data is stored in the database.

b) Select one of:

- **Number of rows** – domain log data that surpasses the specified number of rows is flushed from memory. Enter the desired number of rows adjacent to **Rows**. Disabled by default.
- **Time interval** – domain log data older than the specified time interval is flushed from memory. Enter the desired duration adjacent to **Minutes**. The default is 5.
- **Either rows or time interval** – domain log data is flushed from memory according to whichever value is reached first: either the specified number of rows or the specified time interval. Enter the desired rows and duration adjacent to **Rows** and **Minutes**, respectively.

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

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

Note: By default, the domain log database flushes data every 5 minutes. Alternatively, you can flush data immediately by removing or decreasing the default values, but doing so impacts performance.

8. Optional. To change the data source, select an available database from the **Domain log database endpoint** drop down list.

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

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

10. Click **OK**.

Reviewing Domain Log Data

An administrator reviews logged data by creating log filters. The filters enable you to retrieve data logged for a specific thread, application, user, connection, among other options.

You can retrieve log data without using any filters, however, when there is large number of activities being logged, it may be advisable to filter the results to a more manageable size by specifying search conditions in the log filter (user, application, or thread-id).

You can combine multiple log filters that are common with sub-system specific filters when viewing in a sub-system view, and combine multiple sub-system filters in the ALL tab to retrieve the data of interest.

Creating Log Filters

Filter the log data by creating filters across subsystems that define the appropriate search criteria.

1. In the left navigation pane of the SAP Control Center, select the **Domains** node.
2. Select the domain node and select the **Log** node.
3. In the right administration pane, select the **General** tab.
4. Select **+** to add a filter definition to a subsystem.
5. In the Filter Definition dialog, enter the **Name** and **Description** of the filter.
6. Select the **Sub System**.
7. Select the filter criteria and assign values to the criteria selected. You can use the logical operations to compose the criteria.

   Note: You use the 'AND' logical operator to highlight filter relations belonging to the same subsystem. Filter definitions among multiple subsystems use the 'OR' logical operator.
8. Click **OK**.

Reusable Log Filters

Create reusable log filters that you can use as a base. One strategy is to create a base log filter for each of the supported log subsystems, and for significant categories within subsystems. Another strategy is to create common log filters (useful across subsystems) on specific criteria, such as thread ID, user, package, and so forth.

You can modify these base log filters as needed for more specific searches, or clone the log filter and modify it for a specific search.

Using End to End Tracing to Troubleshoot and Diagnose Device Problems

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

Prerequisites

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

Task

1. The System Administrator creates a logging profile for a given application connection ID based on the users information.

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

   The transaction id and root context id fields are used to trace a given request. The field values should be saved in all modules where trace data is required. Each request made to the server has a different transaction id and all requests within a given session have the

same root context id. Each such session has different root context ids. For example, consider the following snippet of client code:

```
E2ETraceTestDB.Synchronize();
int id = 121;
Customer customer = new Customer();
customer.Fname = "Ritchie";
customer.Lname = "Dennis";
customer.Id = id;
customer.City = "CA";
customer.Save();
customer.SubmitPending();
E2ETraceTestDB.Synchronize();
```

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

Table 16. Transaction and Root Context ID Logging

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

8. View the server log. The trace level set in the passport by the client devices determines the amount of end to end tracing captured in these logs:
   - Domain logs
   - Payload logs
   - Performance logs
   - SAP Mobile Platform log level

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

Table 17. Trace Levels

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

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

Agentry Server Logs

Each Agentry application has its own Agentry Server instance that runs on the SAP Mobile Server node. Some Agentry Server instance logs can be configured through SAP Control Center. Agentry application message and event logs can be viewed through SAP Control Center.

Agentry Server Logs Overview

The Agentry Server can generate several log messages that are written to one or more log files. There are different categories of log file messages that pertain to a different aspect of the application:

- Client-Server communications
- System connection communications
- User-specific information for both Client-Server and system connection synchronization
- Definition loading and processing
- Code file (SQL, Java, batch files, etc.) processing, including post-SDML expansion and results, JVM messages, and so on
- Agentry Server system log files (these log files are always generated by the server and have no configurable behaviors)

You can use the log messages generated by the Agentry Server to diagnose development or production issues, determine processing times for numerous aspects of synchronization, and to obtain various other pieces of information.

Agentry Server System Log Files

Following are the system log files for the Agentry Server:

- `messages.log*`
- `events.log*`
- `startup.log`
- `shutdown.log`

** - These log files can be viewed using the SAP Control Center.*

These log files are always generated and cannot be disabled. Their contents do not have different verbosity levels.

Log Message Contents and Categories

Log files are enabled or disabled, configured, and managed using the SAP Control Center. The options available are to enable or disable the various log files, to specify whether log files are generated based on a processing thread or more narrow areas of functionality, and to set the verbosity level of the log message contents.

The log messages generated by the Agentry Server are categorized as follows:

- **Server:** Messages generated as a result of Agentry Server processing. The contents are an inclusive set of the other categories, with the messages including all those related to any processing performed by the Agentry Server.
- **User:** Messages generated that are specific to a single user. The contents are an inclusive set of the other categories, with the messages including all those related to any user-specific processing performed by the Agentry Server.
- **System Connection:** Messages generated resulting from processing performed by the Agentry Server for back end data synchronization for a specific system connection. The contents are messages that result from processing the various synchronization definitions within the application. Each system connection within an application has its own set of log messages that you can individually enable and configure.
- **Client Communications:** Messages generated resulting from Client-Server communications. These messages include listings of involved IP addresses and port numbers, opening of sockets, transfer of data in both directions, and other messages sent between the Agentry Client and Agentry Server.

The contents of all log messages, regardless of category, include both error messages and informational or status messages. Each message includes a date and time stamp precise to the millisecond. The verbosity of the messages generated is configured when the category of log messages are enabled.

Log Messages Versus Log Files

Log messages generated by the Agentry Server are stored in *log files*. When enabling and configuring the logging behavior of the Agentry Server, the contents of the log messages are configured. In addition, the files into which those messages are written is also configured. By

default, when a log message category is enabled, those messages are written to a log file for a processing thread. Each Agentry Server processing thread will have its own log file containing all messages generated by the Agentry Server for processing within that thread, and for the category of log messages that have been enabled.

As an optional additional method for storing log messages, each category of log messages can be written to a category-specific log file. When this is enabled, all log messages generated by the Agentry Server for a specific category are written to the same log file, regardless of processing thread. The messages within the category-specific log files are grouped by the processing thread. Therefore, a category log file contains log messages resulting from processing related to that category.

Following are the individual log files that can be generated for each category, in addition to the thread log files:

- **Server category:** `Server-`*`ServerName`*`.log` where *`ServerName`* is the configured name for the Agentry Server in the `Agentry - systemConnection` configuration option. The category is `Agentry` by default.
- **User category:** `User-`*`UserID`*`.log` where *`UserID`* is the user's Agentry Client login ID.
- **System Connection category:** `BackEnd-`*`SystemConnection`*`.log` where *`SystemConnection`* is the name of the system connection section (i.e., `SQL-1`, `Java-2`).
- **Client Communications category:** `FrontEnd-`*`FrontEndType`*`.log` and `ANGEL Connection-`*`www.xxx.yyy.zzz`*`.log` where *`FrontEndType`* is the connect type (i.e., `ANGEL Front End`, `Web Front End`). The second file is specific to the ANGEL Connection type. One file is created for each Agentry Client where *`www.xxx.yyy.zzz`* is the Client's IP address as seen by the Agentry Server.

Configuring the log files to generate separately based on the log message category will have a negative impact on Agentry Server performance, as multiple processing threads are logging messages to the same physical log file. Only a single message is written at a time to such a log file. Therefore, each thread must wait until the log file is available before it can write its log message to the log file. It is recommended to configure log files in this manner only for Agentry Development Servers with any regularity. It can be done in a production environment, but only when necessary and for limited durations.

All log files are written to the applicable Agentry application folder in `C:\SAP \MobilePlatform\Servers\UnwiredServer\logs`. Log files are rolled periodically by the Agentry Server, during Agentry Server startup or restart, and on command. The files are rolled to the sub-directory `Logs-Rolled`.

Configuring Agentry Server Instance Logs

Each Agentry application has its own Agentry Server instance that runs on the SAP Mobile Server node. The Agentry Server instance can generate several log messages that are written to one or more log files. You specify the level of verbosity for the log messages and whether the

messages should be written to a separate log file. You can view the log files for an Agentry Server instance in *SMP_HOME*\Servers\UnwiredServer\logs\<*agentryApplicationID*>.

1. In the navigation pane of SAP Control Center, expand the **Applications** node and select the application.
2. In the administration pane, click the **Logs** tab.
3. Select the log for which to configure settings.
4. Click the **Level** for the selected log.
5. Select a verbosity level for the generated log messages.
6. To record the messages to a separate file, in addition to the thread files, click the **Separate File** for the selected log record and select **Yes**.
7. Click **Save**.

SNMP Notifications

(Not applicable to Online Data Proxy) You can set up SAP Control Center *X.X* to include a Simple Network Management Protocol (SNMP) plug-in that sends notifications to the configured target when the state of an SAP Mobile Server changes (that is, from running to stopped, or from stopped to running).

SNMP is the standard protocol for managing networks and exchanging messages. If the SNMP plug-in is set up for a SAP Control Center *X.X*, the plug-in creates notifications in response to predetermined status change events that are detected and signaled by the SAP Mobile Server code. When the plug-in generates a notification, a single copy of the notification is transmitted to each target. The SNMP notification target must be the host name and port of a network monitoring station (NMS) that has the ability to process SNMP notifications; SAP Mobile Platform does not include this functionality. Targets and other notification configuration information are read when the SAP Control Center *X.X* is initialized; therefore, you must stop and restart the agent when enabling SNMP.

Setting Up SNMP Notifications

Setting up SNMP notifications requires you to modify the configuration for SAP Control Center *X.X* and correctly configure an existing SNMP network monitoring station (NMS). Always test the implementation to validate its setup.

1. *Enabling SNMP Notifications for SAP Mobile Platform*

   Enable the SAP Mobile Platform SNMP plug-in for SAP Control Center and set up a notification target to receive messages when the status of a local SAP Mobile Server changes.
2. *Handling Transmitted SNMP Notifications*

Configure an SNMP notification handling tool to process SAP Mobile Platform SNMP notifications.

3. *Testing Notifications with SNMP Queries*

   Test the configuration of the SAP Mobile Platform SNMP plug-in for SAP Control Center by using a management information base (MIB) browser tool to execute an SNMP query.

Enabling SNMP Notifications for SAP Mobile Platform

Enable the SAP Mobile Platform SNMP plug-in for SAP Control Center and set up a notification target to receive messages when the status of a local SAP Mobile Server changes.

Prerequisites

Before modifying the SNMP `agent-plugin.xml` and `service-config.xml` files, stop the SAP Control Center *X.X* service.

Task

Note: The SNMP plug-in for SAP Control Center detects the status of only the SAP Mobile Server running on the same host computer as your instance of SAP Control Center.

1. Stop the SAP Control Center service.
2. Enable the SNMP plug-in to run when SAP Control Center starts:
   a) Open *SCC_HOME*`\plugins\com.sybase.smpsnmpplugin_X.X.X\agent-plugin.xml`.
   b) Set `register-on-startup="true"`.
   c) (Optional) Modify the value of the `sup.server.ping.schedule.interval` property to specify how often (in seconds) the SNMP plug-in pings SAP Mobile Server to detect the server status. The default is 100.
3. (Optional) Change the SNMP notification target to a custom destination:
   a) Open *SCC_HOME*`\services\Snmp\service-config.xml`.
   b) Modify the `snmp.notification.targets` property as follows:

      ```
      set-property property="snmp.notification.targets"
      value=<hostname or IP>/<port number>
      ```

      For example, `set-property property="snmp.notification.targets" value="127.0.0.1/162,10.42.33.136/49152".<hostname or IP>` indicates the server where the network monitoring station (NMS) is located. `<port number>` specifies the SNMP notification port of the NMS. The default SNMP notification port is 162. As indicated in the example, you can set multiple SNMP notification targets.
4. Start the SAP Control Center service.

Handling Transmitted SNMP Notifications

Configure an SNMP notification handling tool to process SAP Mobile Platform SNMP notifications.

Prerequisites

Install an SNMP notification handling tool, such as HP OpenView.

Task

1. On the SNMP notification target host computer, launch an SNMP notification handling tool.
2. Using the third-party documentation, configure the SNMP notification handling tool to process SAP Mobile Platform SNMP notifications. Configure the notification handler to listen for notifications on the port specified in the "snmp.notification.targets" property of the *SCC_HOME*\services\Snmp\service-config.xml file.

Testing Notifications with SNMP Queries

Test the configuration of the SAP Mobile Platform SNMP plug-in for SAP Control Center by using a management information base (MIB) browser tool to execute an SNMP query.

Prerequisites

Install a third-party MIB browser tool.

Task

1. Load *SCC_HOME*\plugins\ com.sybase.smpsnmpplugin_X.X.X \SYBASE-SUP-MIB.txt as a module into your MIB browser.
2. Configure the MIB browser settings to use an SNMPv3 information module with the parameters specified in the *SCC_HOME*\services\Snmp\service-config.xml file:

Property name	Description	Default value
snmp.transport.mappings	SNMP agent host/port	UDP (0.0.0.0/1498) The host "0.0.0.0" represents the local host. Changing this value to a different IP or host name causes the service to fail, since the SNMP service functions only on the local host. The default port number is 1498. You can set a different port for SNMP queries, however, you must ensure that it is not occupied by another SNMP agent.
snmp.usm.user	User name	snmpadmin
snmp.auth.passphrase	Auth password	Sybase4me

3. In the MIB browser, set the **Auth Protocol** to `SHA` and the **Security Level** to `Auth,NoPriv`.
4. In the object identifier tree of the MIB browser, navigate to `SYBASE-MIB \enterprises\sybase\sup\supObjects\supStatusTable \supStatusEntry` and select **get SNMP variable**.
   SAP Mobile Server status information appears in the data console.

Using MLMon to Track Synchronization Events

MLMon is an administration utility that enables you to monitor synchronization performance by capturing all the details of a synchronization event .

For an overview of MLMon, see *http://dcx.sybase.com/index.html#1201/en/mlserver/ml-monitor.html*. For information on the MLMon user interface, see *http://dcx.sybase.com/index.html#1201/en/mlserver/using-ml-monitor.html*.

For information on how to use MLMon with SAP Mobile Server, see *Creating a MLMon User for SAP Mobile Server* and *Synchronization Monitor (mlmon) Utility*.

1. *Creating a MLMon User for SAP Mobile Server*

   Before you can use the MLMon synchronization utility, you must create a user to connect to the SAP Mobile Server server.
2. *Synchronization Monitor (mlmon) Utility*

   Monitors the progress of replication synchronization and checks for potential data contention or bottlenecks. This utility is located in SMP_HOME \Servers \SQLAnywhereXX\BIN32.

Creating a MLMon User for SAP Mobile Server

Before you can use the MLMon synchronization utility, you must create a user to connect to the SAP Mobile Server server.

1. Navigate to *SMP_HOME*\Servers\SQLAnywhere*XX*\BIN*XX*
2. Add the MLMon user to the cache database by running this command:

```
mluser -c
"DSN=<databaseSourceName>;UID=<dbaUserID>;PWD=<databasePassword>"
-u <mobilinkUserName> -p <mobilinkPassword>
```

Note: Default values for this command are:

- DSN=default-cdb
- UID=dba
- PWD=SQL

When the user has been added you will see a message like the following:

```
Server User Utility Version 12.0.1.3519
This software is using security technology from Certicon Corp.
User:MobilinkUserName
```

You can now start the MLMon synchronization utility. For more information, see *Synchronization Monitor (mlmon) Utility*.

For more information on working with MLMon users, see *http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.sqlanywhere.12.0.1/mlserver/dbmluser-ml-ref.html*.

Synchronization Monitor (mlmon) Utility

Monitors the progress of replication synchronization and checks for potential data contention or bottlenecks. This utility is located in *SMP_HOME* \Servers\SQLAnywhere*XX* \BIN32.

Prerequisites: To use this utility, ensure you are using the correct version of the JDK. For details, see *SAP Mobile Platform Runtime Requirements* in *Supported Hardware and Software*. If you choose to use an existing JDK option during installation, also ensure that you have set the JAVA_HOME environment variable to the installation location of this JDK version. Otherwise, you must use a text editor to modify the existing JDK path in the batch file itself.

Syntax

```
mlmon [connect-options|inputfile.{mlm|csv}]
```

When you execute this command, you are prompted to provide:

- Valid administrator credentials
- Replication protocol (default: HTTP)
- Replication port (default: 2480)

Parameters

- **connect-options –** allows you to connect to the SAP Mobile Server on startup. A monitor connection starts like a synchronization connection to the SAP Mobile Server. Allowed values are:

Option	Description
`-u ml_username`	Required to connect to the SAP Mobile Server.
`-p password`	Required to connect to the SAP Mobile Server.

`-x [tcpip | tls | http | https] [(keyword=value;...)]`	Required to connect to the SAP Mobile Server. The keyword=value pairs can be:
	• Host – the network name or IP address of the computer where the SAP Mobile Server is running. By default, it is the computer where the monitor is running.
	• Protocol – should be set to the same network protocol and port as the SAP Mobile Server is using for synchronization requests.
	• Additional Network Parameters – optional parameters, including:
	  • `buffer_size=number`
	  • `client_port=nnnn`
	  • `client_port=nnnn-mmmmm`
	  • `persistent=[0|1]` (HTTP and HTTPS only)
	  • `proxy_host=proxy_hostname` (HTTP and HTTPS only)
	  • `proxy_port=proxy_portnumber` (HTTP and HTTPS only)
	  • `url_suffix=suffix` (HTTP and HTTPS only)
	  • `version=HTTP-version-number` (HTTP and HTTPS only)

`-o outputfile.{mlm|csv}`	Closes the monitor at the end of the connection and saves the session in the specified file.

- **inputfile.{mlm|csv} –** prompts the monitor to open the specified file.

SAP Solution Manager

SAP Solution Manager provides tools that you can use to manage SAP Mobile Platform as part of your overall SAP landscape.

Use SAP Control Center to configure a connection from SAP Mobile Server to SAP Solution Manager. Once connected, you can use the Solution Manager interface to view change records, as well as perform end-to-end traces, and exception and workload analysis.

For more detailed information about configuring SAP Solution Manager to be used in conjunction with SAP Mobile Platform, see *Maintenance of SAP Mobile Platform in the System Landscape*.

For complete SAP Solution Manager documentation, see *SAP Solution Manager Setup*.

Configuring SAP Solution Manager URL

Define and maintain a URL definition associated with an SAP Solution Manager instance for each application in the landscape. This endpoint is used to upload the business transaction XML generated by the client device platforms in an end-to-end trace session.

1. In the left navigation pane, select **Configuration**.
2. In the right administration pane, click the **General** tab.
3. From the menu bar, select **Components**.
4. Select **Solution Manager** and click **Properties**.
5. In the Solution Manager Component Property dialog, enter the URL associated with the appropriate SAP Solution manager.

Workload Analysis with Wily Introscope Agent

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

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

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

Enable the appropriate agent to ensure the proper data is tracked and uploaded to the Introscope tool.

Configuring and Enabling Introscope Agents for an SAP Mobile Server

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

Prerequisites

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

Task

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

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

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

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

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

3. To stop communicating KPIs, disable the agent:

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

Key Performance Indicators

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

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

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

KPI Grouping	Description	Example
Authentication	Enumerates user authentication performed against a single login module. The KPI is a direct reflection of the execution time for the authentication attempt against the module type used. If a security configuration contains multiple login modules, a single user authentication result may occur in more than a one event.	`SAP Mobile Platform|Security|{SecurityConfigurationName}|Authentication|{LoginModuleType}`
Registration		
	Enumerates all automatic application connection registration events.	`SAP Mobile Platform|Applications|Registration`
	Average time taken for registering a user through the message channel.	`SAP Mobile Platform|Applications|Registration|MessageChannel`
	Average time taken for registering a user through the HTTP channel.	`SAP Mobile Platform|Applications|Registration|HttpChannel`
	Average time taken for registering a user through the JMO channel.	`SAP Mobile Platform|Applications|Registration|JmoChannel`
Replication Events		

KPI Grouping	Description	Example
	The average time uploading data from the client application to the cache. This KPI does not include network time.	`SAP Mobile Platform|{domain}|Cache|{package}|Synchronization|UploadData`
	The time processing entity operations in the package until the download starts.	`SAP Mobile Platform|{domain}|Cache|{package}|Synchronization|PrepareForDownload`
	The amount of time fetching rows which need to be downloaded from the cache. This does not include network time.	`SAP Mobile Platform|{domain}|Cache|{package}|Synchronization|DownloadData`
	The average time processing messages for asynchronous cache operations and putting the message on the messaging sub system queue for transport. This does not include network time and the time spent by messaging subsystem.	`SAP Mobile Platform|Cache|ProcessReplayBatchMessage`
	The event of generating internal data change event notifications.	SAP Mobile Platform|{domain}|Cache|{package}|ScheduledNotify
	The average time generating internal data change event notifications.	`SAP Mobile Platform|Cache|{classname}|{method}|Cache|{classname}|{method}`
Push/Server-initiated Synchronization (SIS)		

KPI Grouping	Description	Example
	Determines which application connections are affected by push changes since the last run of this task.	`SAP Mobile Platform|{domain}|Cache|{package}|ScheduledNotify`
	The average time creating SIS notifications.	`SAP Mobile Platform|{domain}|Cache|{package}|ScheduledNotify|CreateNotifications`
	The average time creating a single SIS notification.	`SAP Mobile Platform|{domain}|Cache|{package}|ScheduledNotify|CreateOneNotification`
	The average time taken for SAP Mobile Server to send notifications to clients over the messaging channel.	`SAP Mobile Platform|{domain}|Cache|{package}|ScheduledNotify|InternalTransport`
	The average time taken for SAP Mobile Server to send notifications to clients over the HTTP channel	`SAP Mobile Platform|{domain}|Cache|{package}|ScheduledNotify|HttpTransport`
Messaging Synchronization		
	The average time processing all incoming asynchronous messages from messaging applications.	`SAP Mobile Platform|Cache|Messaging|ProcessIncomingMessage`
	The average time processing messages related to bulk subscribe.	`SAP Mobile Platform|Cache|Messaging|ProcessBulkSubscribeMessage`
	The average time determining which application connections are affected by changes since the last run of this task.	`SAP Mobile Platform|{domain}|Cache|Messaging|{package}|ScheduledPush`

KPI Grouping	Description	Example
	The average time spent in background workerthread processing "persistent" events that will lead to import message generation for cache messaging based applications.	`SAP Mobile Platform|Cache|Messaging|GenerateOutgoingMessages`
	The average time producing the actual push job (GenerateOutgoingMessages) to run in the context of a particular application connection for which data is to be delivered.	`SAP Mobile Platform|{domain}|Cache|Messaging|{package}|ScheduledPush|CreatePushJobs`
Data Services (EIS read or write)		
	The average execution time for a single EIS operation of an entity.	`SAP Mobile Platform|{domain}|Cache|{package}|{mbo}|{operation}`
	The time spent modifying the cache with the supplied rows of an entity.	`SAP Mobile Platform|{domain}|Cache|{package}|{mbo}|ModifyCache`

KPI Grouping	Description	Example
	Determines the degree of change. The merge processor only updates the cache incrementally when the EIS partition is empty, or the cache partition is empty and there are no foreign keys or the cache is unpartitioned. Otherwise, the change commands are collected and executed in series as a batch when applyDeltas() is invoked.	`SAP Mobile Platform|{domain}|Cache|{package}|{mbo}|DetermineDelta`
	For the specified partition, this method compares the contents of the cache with the contents of the partition in the EIS; any changes are detected by computeDeltas() and translated into change commands.	`SAP Mobile Platform|{domain}|Cache|{package}|{mbo}|ApplyDelta`
Data maintenance		
	The average time spent removing logically deleted rows in the cache that are older than the oldest synchronization time on record in the system. This cleanup task also removes unused or stale partitions.	`SAP Mobile Platform|{domain}|Cache|{package}|Data Maintenance|PurgeEntities`

KPI Grouping	Description	Example
	The average time spent removing client log records that have already been synchronized to the device, or are no longer associated with active users.	`SAP Mobile Platform|{domain}|Cache|{package}|Data Maintenance|PurgeClientLog`
	The average time spent removing historical data on MBO data refresh and operation replay failures, which result from system or application failures.	`SAP Mobile Platform|{domain}|Cache|{package}|Data Maintenance|PurgeErrorLog`
	The average time spent in removing subscriptions that are not active for the 'number of inactive days' in the schedule task configuration.	`SAP Mobile Platform|{domain}|Cache|{package}|Data Maintenance|PurgeSubscriptions`
DCN/WFDCN incoming events from EIS for cache manipulation or HWA trigger		
	The average time spent in parsing and processing the DCN request for Hybrid Apps on SAP Mobile Server.	`SAP Mobile Platform|{domain}|Hybrid App Notification|ProcessRequest`
	The average time spent in parsing and processing the DCN request on SAP Mobile Server.	`SAP Mobile Platform|{domain}|Cache|{package}|Data Change Notification|ProcessRequest`
	Processing the URL authenticated Hybrid App DCN request on SAP Mobile Server.	`SAP Mobile Platform|{domain}|Hybrid App Notification|POST`

KPI Grouping	Description	Example
	Processing the basic authenticated Hybrid App DCN request on SAP Mobile Server.	`SAP Mobile Platform|{domain}|Hybrid App Notification|HttpPOST`
	Processing the URL authenticated DCN request on SAP Mobile Server.	`SAP Mobile Platform|{domain}|Cache|{package}|Data Change Notification|POST`
	Processing the basic authenticated DCN request on SAP Mobile Server.	`SAP Mobile Platform|{domain}|Cache|{package}|Data Change Notification|HttpPOST`
Messaging		
	The average time processing incoming messages over IIOP.	`SAP Mobile Platform|MessageChannel|Transport`
	The average time executing commands by a registered Handler.	`SAP Mobile Platform|MessageChannel|Dispatcher`
	The event of a single MBO request invocation from a Hybrid App.	`SAP Mobile Platform|MessageChannel|MboRequestHandler`
	The event of requesting a specific customization resource bundle from SAP Mobile Server.	`SAP Mobile Platform|MessageChannel|CustomizationResourceHandler`
	The event of uploading a business transaction in XML from the device to SAP Mobile Server, including the actual time to transfer the file to the Solution Manager.	`SAP Mobile Platform|MessageChannel|BtxUploadHandler`
	The average time inserting the device logs into the domain log database.	`SAP Mobile Platform|MessageChannel|ClientLogsHandler`

KPI Grouping	Description	Example
	The event of logging a trace record in domain log database by messaging sub system.	`SAP Mobile Platform|Message-Channel|E2eTraceHandler`
	Average time spent inserting messages into the messaging queue. This KPI gets called from Push.	`SAP Mobile Platform|Message-Channel|SendDirect`
JMO Messaging		
	Average time processing incoming messages through message channel over JMO path.	`SAP Mobile Platform|Proxy|Jmo-MessageChannel`
Online Data Proxy		
	The time spent handling a single client call from an application in the messaging channel.	`SAP Mobile Platform|Proxy|MessageChannel`
	The time interacting with the backend in the messaging channel.	`SAP Mobile Platform|Proxy|MessageChannel|EndpointCall` `SAP Mobile Platform|Proxy|MessageChannel|EndpointCall|GatewayUrl`
	The average time processing the response from the backend in the messaging channel.	`SAP Mobile Platform|Proxy|MessageChannel|PrepareResponse`
	The time spent handling a single client call from an application in the HTTP channel.	`SAP Mobile Platform|Proxy|HttpChannel`

KPI Grouping	Description	Example
	Invokes the backend in the HTTP channel.	`SAP Mobile Platform|Proxy|HttpChannel|EndpointCall` `SAP Mobile Platform|Proxy|HttpChannel|EndpointCall|GatewayUrl`
	The average time processing the response from the backend in the HTTP channel.	`SAP Mobile Platform|Proxy|HttpChannel|PrepareResponse`
Native Push Notifications		
	The event of processing the push notification message delivery to native push subsystems.	`SAP Mobile Platform|NativePush|POST`

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

DOE-C		
	Average time spent in all DOE-C incoming/outgoing requests.	`SAP Mobile Platform|DOEC|DOERequestProcessing`
	Average time taken processing an incoming request from DOE server.	`SAP Mobile Platform|DOEC|ProcessDOEResponse`
	Average time spent pushing messages from DOE-C to device queue.	`SAP Mobile Platform|DOEC|Push`

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

KPI Grouping	Description	Example

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

Message channel	Generic method called by device -- performs web request (HTTP) of MMS for a synchronized method call over messaging channel for a particular Handler-Id call with respective domain and security configuration	`SAP Mobile Platform|MessageChannel|Transport|{handler-id}-{domain}-{security configuration}`
Hybrid Apps		
	The event of executing the synchronous request from the Hybrid App.	`SAP Mobile Platform|Hybrid Apps|ExecuteRequest`
	The event of executing the asynchronous request from the Hybrid App.	`SAP Mobile Platform|Hybrid Apps|SubmitResponse`
	The event of executing the object query (if any) and putting the message response into the messaging queue.	`SAP Mobile Platform|Hybrid Apps|ProcessEvent`
Outbound messages from SAP Mobile Server to application		

KPI Grouping	Description	Example
	The event of submitting the message from SAP Mobile Server to the device outbound queue.	`SAP Mobile Platform|Cache|Messaging|MessageFromServer|Process`
	The average time taken for the server to acknowledge the message after it was put into the outbound queue.	`SAP Mobile Platform|Cache|Messaging|MessageFromServer|Acknowledge`

CHAPTER 13 SAP Interoperability

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

Different SAP systems have different methods of data exchange.

SAP SLD Server Overview

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

Table 20. SLD Administration Tasks

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

SLD and SAP Mobile Platform Architecture

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

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

- The primary SAP Mobile Server is responsible for the cluster-level scheduling task. It also gathers local server information and routes data to the cluster database. When the schedule

milestone is reached, the primary server generates and uploads the payload from information held in the cluster database.
- Any secondary SAP Mobile Server automatically gathers local server information and routes data to the cluster database. The administrator need not configure this activity as it is controlled by the cluster-level task.
- The cluster database aggregates and holds information downloaded from all cluster server members.

See also
- *Sharing Cluster Data with the SLD* on page 208

Aggregating and Holding Cluster Data
SAP Mobile Platform uses the cluster database to centralize the collection of all cluster data that becomes the payload to SLD servers.

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

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

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

Sharing Cluster Data with the SLD
Configure a cluster to connect to the System Landscape Directory (SLD) and generate and upload the payload that contains all details of the SAP Mobile Platform system.

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

Task
Choose the upload method you wish to use:

See also
- *SLD and SAP Mobile Platform Architecture* on page 207

Uploading Payloads with SAP Control Center

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

1. *Registering or Reregistering SLD Server Destinations*

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

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

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

Registering or Reregistering SLD Server Destinations

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

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

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

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

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

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

   This registers the SLD destination.

Configuring and Enabling Scheduled Payload Generation and Uploads

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

1. In the navigation pane of SAP Control Center, select the name of the cluster for which you want to schedule an SLD payload upload.
2. From the menu bar of the **System Landscape Directory** page, click **Schedule**.
3. To edit an existing schedule for a selected SLD server, click **Edit**.
   a) Configure the schedule:
      - **Schedule repeat** – select how often the schedule should run. Options are **daily**, **hourly**, **custom**, and **never**.
        - If you select **daily** or **hourly**, specify:
          - **Start date** – select the date and time the automated upload should begin. Use the calendar picker and 24-hour time selector.
          - **End date** – select the date and time the automated upload should end.
          - **Days of the week** – select each day the automated upload schedule should run.
        - Select **custom**, to specify the interval granularity in seconds, minutes, or hours, as well as other date and time parameters.

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

Manually Uploading or Exporting Payloads On-Demand

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

1. In the navigation pane of SAP Control Center, select the name of the cluster for which you want to immediately upload an SLD payload.
2. In the administration pane, click the **System Landscape Directory** tab.
3. From the menu bar of the **System Landscape Directory** page, click **Schedule**.

4. Click **Run Now**.
   The payload generation process begins.
5. Upon completion, review the contents of the payload and choose an action:
   - To export and save the contents to a file as XML, click **Save to File** and choose your file output name and location.
   - To upload the contents, select the target SLD servers and click **Finish**.

Uploading Payloads with Scripts

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

1. *Registering the SLD Server Destination with Scripts*

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

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

Registering the SLD Server Destination with Scripts

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

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

UserName	User name for the SLD server.
Password	The user account password used to authenticate the user name entered. Password and Confirm Password must match for the password to be accepted.
ServerHost	The host name or the IP address of the SLD server.
Port	The HTTP(S) port on which the SLD server is running. Enter a valid port number in the range of 0-65535.
Use https	Indicate if you want a secure connection or not.
Write this information to secure file	If you want to store the encrypted HTTP connection information, enter `y`.

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

Generating and Uploading Payloads with Scripts

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

Prerequisites

- Ensure all SAP SAP Mobile Platform services are currently running before you generate the payload.
- You have defined the JAVA_HOME environment variable. For example, if you have installed SAP Mobile Platform on your system, set the JAVA_HOME as `set JAVA_HOME=SMP_HOME\\JDK1.6.0_16`

Task

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

Login name	SAP SAP Mobile Platform administrator's login name
Login password	SAP SAP Mobile Platform administrator's login password

   If the payload generation is successful, the payload file `SUP_PayLoad.xml` is created in the location `SMP_HOMEMobilePlatform>\Servers\UnwiredServer\SLD \SLDREG\`.
3. To upload the payload to SLD, run the batch file `sendPayLoad.bat`.

SAP License Audit and Application Data Overview

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

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

Sharing Application Data with SAP License Audit

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

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

1. *Generating the SAP Audit Measurement File*

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

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

Generating the SAP Audit Measurement File

Use SAP Control Center to generate an audit measurement file.

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

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

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

Uploading the SAP Audit Measurement File

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

SAP Applications Tracked with SAP License Audit

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

Note: Applications created with SAP Mobile Platform 2.1 or earlier are not counted.

ID	Unit
S001	SAP Mobile Platform User
S005	SAP Employee Lookup Mobile User
S010	SAP Leave Request Mobile User

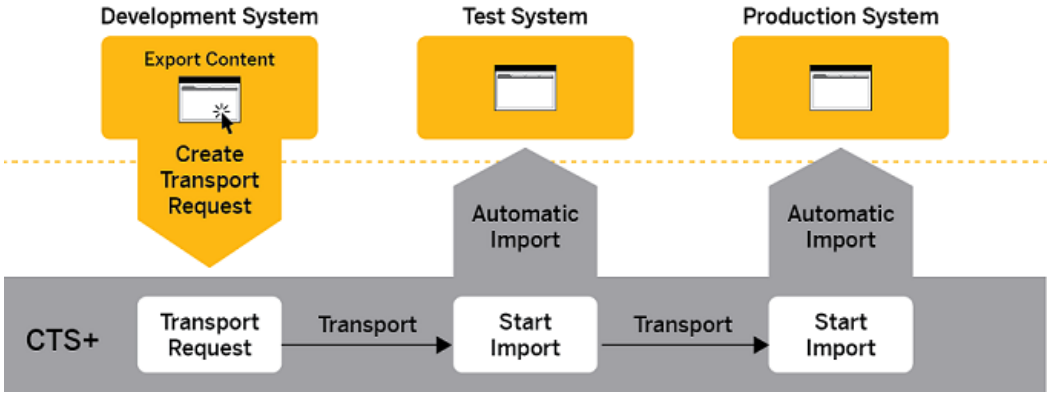
ID	Unit
S015	SAP Travel Receipt Capture Mobile User
S020	SAP Travel Expense Approval Mobile User
S025	SAP HR Approvals Mobile User
S030	SAP Cart Approval Mobile User
S035	SAP Timesheet Mobile User
S040	SAP Sales Order Notification Mobile User
S045	SAP Material Availability Mobile User
S050	SAP ERP Order Status Mobile User
S055	SAP CRM Retail Execution Mobile User
S060	SAP Field Service Mobile User
S065	SAP EAM Work Order Mobile User
S070	SAP ERP Quality Issue Mobile User
S075	SAP Customer and Contacts Mobile User
S080	SAP GRC Access Approver Mobile User
S085	SAP GRC Policy Survey Mobile User
S090	SAP Payment Approvals Mobile User
S095	SAP Customer Financial Fact Sheet Mobile User
S100	SAP Interview Assistant Mobile User
S105	SAP Transport Notification and Status Mobile User
S110	SAP Transport Tendering Mobile User
S115	SAP Manager Insight Mobile User
S120	SAP Electronic Medical Record Mobile User
S125	SAP Sales Mobile User

CTS Overview

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

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

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



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

Table 21. Additional Resources

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

Basic Setup for CTS

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

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

Table 22. CTS Basic Setup

Task	Notes
Create a CTS domain.	If your SAP Solution Manager is not set as the CTS Domain Controller, set it using transaction code STMS. Documentation for creating a CTS domain is on the SAP help portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/44/b4a0b47acc11d1899e0000e829fbbd/frameset.htm*
Configure the Transport Organizer Web UI.	You need to activate services for the Transport Organizer Web UI in order to use it. You need to activate services for the Object List Browser in order to see a detailed list of objects attached to a transport request as part of one file. If CTS is already in use on the SAP Solution Manager where you are doing the configuration, the services should already be activated. Documentation for activating services for the Transport Organizer Web UI is on the SAP help portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/e5/998566c2174196a12b72e7c7af51e7/frameset.htm* **Note:** If you receive error messages when running this application later on, or if you don't want to activate all ICF services, read the error messages carefully and activate the services named in the error messages using transaction SICF.

Task	Notes
Configure the CTS Deploy Web Service.	You need to create an RFC connection for the communication between the AS ABAP and the AS Java of the CTS system. Documentation for configuring the CTS Deploy Web Service is on the SAP help portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/2b/326d6274134cea8b217f24889d19c1/frameset.htm* **Note:** CTSDEPLOY is the standard port that should be used to connect the Deploy Web Service client and the Deploy Web Service. If this port is already in use for other CTS scenarios and cannot be used for SAP Mobile Platform, create an additional port with a different name, for example, CTSDEPLOY_SUP.

See also

- *Configuring CTS* on page 217
- *Transporting Artifacts Between Servers Using CTS* on page 221
- *Troubleshoot CTS Imports* on page 226

Configuring CTS

To configure CTS for transporting SAP Mobile Platform development artifacts between different SAP Mobile Platform system roles, administrators must create an SAP Mobile Platform application type, set up systems and transport routes for the SAP Mobile Platform server environments, and prepare SAP Mobile Platform transport scripts.

Table 23. CTS Configuration Tasks

Task	Notes
Create an application type for SAP Mobile Platform.	If the SAP Mobile Platform application type is already defined in CTS+, you do not need to complete this task.

Task	Notes
Set up source and target systems for the required server environments.	Documentation for creating systems in the Transport Management System is on the SAP Help Portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/en/bf/e4626214504be18b2f1abeeaf4f8e4/frameset.htm* When creating your target system, select **Other** as the deployment method.
Define transport routes to connect your systems.	Documentation for configuring transport routes is on the SAP Help Portal: *http://help.sap.com/saphelp_ctsplug20sm71/helpdata/ en/44/b4a1df7acc11d1899e0000e829fbbd/ frameset.htm*
Prepare the SAP Mobile Platform transport scripts .	

See also

- *Basic Setup for CTS* on page 216
- *Transporting Artifacts Between Servers Using CTS* on page 221
- *Troubleshoot CTS Imports* on page 226

Setting Up the Application Type in CTS

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

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

Field	Value
Application Identifier	`SUP`
Application Description	SAP Mobile Platform objects.

Field	Value
Support Details	Contact SAP support by creating a customer message using component `MOB-SUP`.

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

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

Configuring the Target System in CTS

You must configure the SAP Mobile Platform target system in CTS for transporting SAP Mobile Platform application objects.

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

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

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

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

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

Preparing Transport Scripts for CTS

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

Prerequisites

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

Task

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

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

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

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

Transporting Artifacts Between Servers Using CTS

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

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

Table 24. Administration Tasks

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

See also

- *Basic Setup for CTS* on page 216
- *Configuring CTS* on page 217
- *Troubleshoot CTS Imports* on page 226
- *MBO Package Export Archive* on page 402

- *Hybrid App Export Archive* on page 403
- *Application Export Archive* on page 403

Exporting Packages, Applications and Hybrid Apps

Export MBO packages, applications, and Hybrid Apps using either the SAP Control Center administration console or the command line.

Exporting MBO Packages

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

Prerequisites

Before beginning, review import requirements and best practices.

Task

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

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

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

Next

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

Exporting Applications

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

Prerequisites

Before beginning, review import requirements and best practices.

Task

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

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

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

Next

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

Exporting Hybrid Apps

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

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

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

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

Next

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

Creating a Transport Request in CTS

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

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

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

---

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

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

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

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

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

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

Import Requirements and Best Practices

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

MBO Package Imports

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

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

Application Imports

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

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

Hybrid App Imports

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

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

Import Results

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

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

- Cache group settings

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

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

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

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

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

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

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

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

See also

- *Troubleshoot CTS Imports* on page 226

Troubleshoot CTS Imports

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

Table 25. Import MBO Package Return Codes

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

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

Table 26. Import Hybrid App Return Codes

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

Table 27. Import Application Return Codes

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

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

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

See also

- *Basic Setup for CTS* on page 216
- *Configuring CTS* on page 217

- *Transporting Artifacts Between Servers Using CTS* on page 221

APPENDIX A System Reference

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

It covers the location of crucial system files and file systems, as well as other reference material that you might need when you are administering the SAP Mobile Platform production environment: for example, logging details, configuration properties, and ports, service names, and processes used by each component.

Installation Directories

To ensure a successful installation, review the SAP Mobile Platform server component installation directories.

- The following tables show the high-level directories created in a single-node installation (all SAP Mobile Platform server components installed on a single host).
- In a multi-node or cluster installation, some of these directories are present only on a particular type of host.

By default, SAP Mobile Platform server components are installed in the `C:\Sybase\UnwiredPlatformC:\SAP\MobilePlatform` directory. In this guide, *`SMP_HOME`* represents the SAP Mobile Platform installation directory, down to the `UnwiredPlatformMobilePlatform` folder.

Table 28. SAP Mobile Platform Installation Subdirectories

Directory	Description
`_jvm`	JVM used by the uninstaller.
`sup`*`XX`*`ebflogs`	Log files created each time `installebf.bat` is run. Appears only in EBF installations upgraded from an earlier version of SAP Mobile Platform.
`InstallLogs`	Log files created each time the SAP Mobile Platform Runtime installer is used. Use these logs to trouble-shoot installer issues.
`IntroscopeAgent`	Introscope Agent for 64-bit Installations.

Directory	Description
JDK*x.x.x_x*	JDK required by SAP Mobile Platform components.
sapjco	SAP Java Connector files.
scc_cert	Certificate files for SAP Control Center.
Servers	SAP Mobile Platform server components.
Servers\AgentryServer	Agentry server.
Servers\MessagingServer	SAP messaging server.
Servers\SQLAnywhere*xx*	Database server for cache, cluster, and logging databases. Default database file location is the data\ subdirectory.
Servers\UnwiredServer	SAP Mobile Server components.
Servers\UnwiredServer\doe-c_clu	SAP® Data Orchestration Engine Connector (DOE-C) Command Line Utility components. CLU.bat in bin directory starts the DOE-C console.
Servers\UnwiredServer\doecSvlet	SAP® Data Orchestration Engine Connector (DOE-C) runtime components.
Servers\UnwiredServer\licenses	SySAM license files. When an unserved license is updated, copy the new files here.
sup*XX*upgradesmp*XX*upgrade	Appears only in installations upgraded from an earlier version of SAP Mobile Platform.
ThirdParty	License terms of third-party components included in SAP Mobile Platform.
Uninstallers	Uninstallers for SAP Mobile Platform Runtime components.
Uninstallers\MobilePlatform	SAP Mobile Platform Runtime uninstaller.
Util	Utilities used by the SAP Mobile Platform Runtime installer.

By default, SAP Control Center components are installed in the C:\SAP\SCC-*XX* directory.

Note: If you have other SAP products installed on the same host as SAP Mobile Server, you may have more than one version of SAP Control Center.

Table 29. SAP Control Center Installation Subdirectories

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

Port Number Reference

Change SAP Mobile 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 SAP Control Center port numbers so you do not accidentally use these ports when you change SAP Mobile Platform ports. You can change some SAP Control Center default ports, but, in some cases, you should not.

Note: To make SAP Mobile Server port number changes, temporarily stop the other service consuming those ports. Use SAP Control Center to make the changes, then restart SAP Mobile Server.

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

Port	Description	Default Port	Instructions for Changing
Data tier (CDB) server	Port number for the data tier that manages transactions between the enterprise information system and mobile devices.	5200	Do not change the CDB port.
Management ports	IIOP port number on which the SAP Mobile Server listens for SAP Control Center administration requests.	2000 2001 for secure management (default)	Default is recommended. No change is required.
HTTP ports	HTTP port number on which SAP Mobile Server listens for: • Data change notification (DCN) events (server authentication). • HTTP channel notification (mutual authentication).	8000 for HTTP 8001 for HTTPS	Configure in SAP Control Center by selecting **Configuration**, clicking the **Web Container** tab and entering a new DCN port or secure DCN port, as required. See *Configuring Web Container Properties* in *SAP Control Center for SAP Mobile Platform* online help.

Port	Description	Default Port	Instructions for Changing
Synchronization	Port numbers on which SAP Mobile 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 SAP Control Center by selecting **Configuration**. In the **Components** tab, select **Replication** or **Messaging**, click **Properties** and enter a new synchronization port, as required. **Note:** If there is a conflict for port 2480 or 2481, SAP Mobile Server will not start, and you cannot use SAP Control Center to modify them. To correct the problem, you must temporarily stop the service that uses the conflicting port, then start SAP Mobile Server. For replication payloads, see *Configuring Replication Subscription Properties* in *SAP Control Center for SAP Mobile Platform* online help. For messaging payloads, see *Configuring Messaging Properties* in *SAP Control Center for SAP Mobile Platform* online help.
Messaging server administration	Port number for the messaging service for SAP messaging clients.	5100 for administration services	Cannot be changed in SAP Control Center. Use the *SMP_HOME*`\Servers\Messaging Server\Bin\AdminWebServicesTool.exe` command line tool to change the messaging service Web service port. This tool has built in online help describing how to use the tool. From the command prompt run: *SMP_HOME*`\Servers\Messaging Server\Bin > AdminWebServicesTool.exe set=<`*port*`> restart`

Port	Description	Default Port	Instructions for Changing
SAP 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 SAP Control Center repository database port 8282, 8283 for default Web container ports	• 9999 – default RMI agent port. The port is set in: *SCC_HOME*\services\RMI\service-config.xml • 2100 – default JMS messaging service port. The port is set in: *SCC_HOME*\services\Messaging\service-config.xml • 3638 – default SAP Control Center repository database port. The default port is set in: *SCC_HOME*\services\SccSADataserver\service-config.xml • 8282, 8283 – default Web container ports. The default ports are set in: *SCC_HOME*\services\EmbeddedWebContainer\service-config.xml Before you make any changes to these files, stop SAP Control Center X.X service. Start the service after you complete the changes. If any of the subsystems fail to start, check the SAP Control Center `agent.log` for error messages.
Relay server	Port numbers on which Relay Server listens for requests.	80 for the HTTP port 443 for the HTTPS port	Change the value for the cluster in SAP Control Center for SAP Mobile Platform. You can then generate the file and transfer it to the corresponding SAP Mobile Server host. See *Setting Relay Server General Properties* in *SAP Control Center for SAP Mobile Platform*.

Port	Description	Default Port	Instructions for Changing
Agentry client-server ANGEL port	Port number reserved for Agentry client-server communications using the ANGEL connect type.	7003	For information on network adapter and port configuration options, see the following topics in *System Administration* • *Configuring Agentry Client-Server Communications* • *Agentry Server: [ANGEL Front End] Configuration Section* :
SAP Mobile Platform reserved	Port numbers reserved for internal use by SAP Mobile Platform components	2638 4343 6001 5500 8002 27000 5701/5702 are for `mlsrv12.exe` 5011 is for `OB-MO.exe`	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 SAP Mobile 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.

SAP Mobile Platform Windows Services

SAP Mobile Platform Windows services run automatically on your host, with many starting up when the host computer is started or when the installation process finishes. Determine what services exist for each runtime component and what dependencies exist among these services.

Depending on the components installed in the cluster you are administering, some of these services may not appear in your list of Windows services; certain data and server components may be installed on different nodes to facilitate redundancy.

Note: SAP recommends that you only manually start and stop SAP Mobile Platform services for debugging and troubleshooting purposes.

If you are routinely starting and stopping SAP Mobile Server, you should use SAP Control Center for that purpose. SAP Control Center allows you to manage local and remote servers from a single location, and is more efficient than starting and stopping with services or desktop shortcuts.

Component	Service	Description	Dependencies
SAP Mobile Server	SAP Mobile Platform CacheDB	The main, or cache, data tier service.	The SAP Mobile Platform CacheDB service must be started before the SAP Mobile Server service can be started.
	SAP Mobile Platform SampleDB	The SAP Mobile Platform SampleDB.	None.
	SAP Mobile Server	SAP Mobile Server service.	Depends on the SAP Mobile Platform CacheDB service startup and relay server service (if used).
Runtime Databases	SAP Mobile Platform ClusterDB	The database server that manages the data that supports the operation of the cluster.	This service only appears with the data tier installed on its own server in a cluster; generally it should be started and stopped with the SAP Mobile Platform CacheDB service. **Note:** A single-node installation runs these databases in the same service (SAP Mobile Platform CacheDB).
	SAP Mobile Platform LogDataDB	The database server that manages logging for SAP Mobile Platform.	This service only appears with the data tier installed on its own server in a cluster; generally it should be started and stopped with the SAP Mobile Platform CacheDB service.

Component	Service	Description	Dependencies
SAP Control Center	SAP Control Center *X.X*	Provides runtime services to manage, monitor, and control distributed SAP resources. The agent must be running for SAP Control Center to run.	None.

Processes Reference

SAP Mobile Platform Windows processes vary, depending on your components and license type.

Use this table to determine existing processes for each runtime component.

Component	Service	Processes
SAP Mobile Server	Replication synchronization services (via MobiLink)	Cache database: `dbsrv`*`XX`*`.exe` Synchronization on application server: `mlsrv`*`XX`*`.exe`
	Messaging synchronization services	`AdminWebServices.exe`, `ampservice.exe`, `JMSBridge.exe`, `LBManager.exe`, `OBMO.exe`, `OBServiceManager.exe`
	Scale-out node services	Data services: `java.exe`
	Starts MMS service	`mlsrvwrapper.exe`
Relay Server	Relay Server	`rshost.exe`
	RSOE	`rsoe.exe`
SAP Control Center	SAP Control Center *X.X*	`sccservice.exe`
	SAP Control Center repository database	`dbsrv`*`XX`*`.exe`

EIS Data Source Connection Properties Reference

Name and configure connection properties when you create connection pools in SAP Control Center to enterprise information systems (EIS) .

See also

- *Viewing and Editing EIS Connection Properties* on page 71
- *Chapter 7, EIS Connection Management* on page 69

JDBC Properties

Configure Java Database Connectivity (JDBC) connection properties.

This list of properties can be used by all datasource types. SAP does not document native properties used only by a single driver. However, you can also use native driver properties, naming them using this syntax:

```
jdbc:<NativeConnPropName>=<Value>
```

Note: If SAP Mobile Server is connecting to a database with a JDBC driver, ensure you have copied required JAR files to correct locations.

Name	Description	Supported values
After Insert	Changes the value to `into` if a database requires `insert into` rather than the abbreviated `into`.	`into`
Batch Delimiter	Sets a delimiter, for example, a semicolon, that can be used to separate multiple SQL statements within a statement batch.	`<delimiter>`
Blob Updater	Specifies the name of a class that can be used to update database BLOB (long binary) objects when the BLOB size is greater than psMaximumBlobLength.	`<class name>` The class must implement the `com.sybase.djc.sql.BlobUpdater` interface.
Clob Updater	Specifies the name of a class that can be used to update database CLOB (long string) objects when the CLOB size is greater than psMaximumClobLength.	`<class name>` The class must implement the `com.sybase.djc.sql.ClobUpdater` interface.

Name	Description	Supported values
Code Set	Specifies how to represent a repertoire of characters by setting the value of CS_SYB_CHARSET for this datasource. Used when the data in the datasource is localized. If you do not specify the correct code set, characters may be rendered incorrectly.	`[server]` If the value is server, the value of the current application server's defaultCodeSet property is used.
Commit Protocol	Specifies how SAP Mobile Server handles connections for a datasource at commit time, specifically when a single transaction requires data from multiple endpoints. If you use XA, the recovery log is stored in the tx_manager datasource, and its commit protocol must be optimistic. If tx_manager is aliased to another datasource (that is, one that is defined with the aliasFor property), the commit protocol for that datasource must be optimistic. A last-resource optimization ensures full conformance with the XA specification. The commit protocol for all other datasources should be XA_2PC. Alternately, a transaction that accesses multiple datasources for which the commit protocols are optimistic is permitted.	`[optimistic| pessimistic | XA_2PC]` Choose only one of these protocols: • Optimistic – enables connections to be committed without regard for other connections enlisted in the transaction, assuming that the transaction is not marked for rollback and will successfully commit on all resources. Note: if a transaction accesses multiple data sources with commit protocol of "optimistic", atomicity is not guaranteed. • Pessimistic – specifies that you do not expect any multi-resource transactions. An exception will be thrown (and transaction rolled back) if any attempt is made to use more than one "pessimistic" data source in the same transaction. • XA_2PC – specifies use of the XA two phase commit protocol. If you are using two phase commit, then the recovery log is stored in the "tx_manager" data source, and that data source (or the one it is aliased to) must have the commit protocol of "optimistic" or "pessimistic". All other data sources for which atomicity must be ensured should have the "XA_2PC" commit protocol.

Name	Description	Supported values
Datasource Class	Sets the class that implements the JDBC datasource. Use this property (along with the driverClass property) only if you do not have a predefined database-type entry in SAP Mobile Server for the kind of SQL database you are connecting to. For example, you must use this property for MySQL database connections. You can implement a datasource class to work with a distributed transaction environment. Because SAP Mobile Server supports distributed transactions, some datasources may require that a datasource class be implemented for SAP Mobile Server to interact with it. For two-phase transactions, use the xaDataSourceClass connection property instead.	`<com.`*`mydata-source`*`.jdbc.Driver>`
Database Command Echo	Echoes a database command to both the console window and the server log file. Use this property to immediately see and record the status or outcome of database commands. When you enable this property, SAP Mobile Server echoes every SQL query to `ml.log`, which may help you debug your application.	`[true|false]` Set a value of 1 to echo the database commands like `databaseStartCommand`, and `databaseStopCommand`. Otherwise, do not set this property, or use a value of 0 to disable the echo.

Name	Description	Supported values
Database Create Command	Specifies the operating system command used to create the database for this datasource. If this command is defined and the file referenced by ${databaseFile} does not exist, the command is run to create the database when an application component attempts to obtain the first connection from the connection pool for this datasource.	`<command>` Example: `SMP_HOME\Servers\SQLAnywhere11\BIN32\dbinit -q ${databaseFile}`
Database File	Indicates the database file to load when connecting to a datasource. Use this property when the path to the database file differs from the one normally used by the database server. If the database you want to connect to is already running, use the databaseName connection parameter.	`<string>` Supply a complete path and file name. The database file you specify must be on the same host as the server.

Name	Description	Supported values
Database Name	Identifies a loaded database with which to establish a connection, when connecting to a datasource. Set a database name, so you can refer to the database by name in other property definitions for a datasource. If the database to connect to is not already running, use the databaseFile connection parameter so the database can be started. **Note:** For SAP Mobile Server, you typically do not need to use this property. Usually, when you start a database on a server, the database is assigned a name. The mechanism by which this occurs varies. An administrator can use the DBN option to set a unique name, or the server may use the base of the file name with the extension and path removed.	`[DBN|default]` If you set this property to default, the name is obtained from the DBN option set by the database administrator. If no value is used, the database name is inherited from the database type.
Database Start Command	Specifies the operating system command used to start the database for this datasource. If this command is defined and the database is not running, the command is run to start the database when the datasource is activated.	`<command>` Example: `SMP_HOME\Servers\SQLAnywhere11\BIN32\dbsrv`*`XX`*`.exe`

Database Stop Command	Specifies the operating system command used to stop the database for this datasource. If this property is defined and the database is running, this command executes during shutdown.	`<command>`
		
		For a SQL Anywhere® database, where the user name and password are the defaults (dba and sql), enter:
		
		`SMP_HOME\Servers\SQLAny-where11\BIN32\dbsrv`*`XX`*`.exe`

Database Type	Specifies the database type.	`<database type>`

Name	Description	Supported values
Database URL	Sets the JDBC URL for connecting to the database if the datasource requires an Internet connection. Typically, the server attempts to construct the database URL from the various connection properties you specify (for example, portNumber, databaseName). However, because some drivers require a special or unique URL syntax, this property allows you to override the server defaults and instead provide explicit values for this URL.	`<JDBCurl>` The database URL is JDBC driver vendor-specific. For details, refer to the driver vendor's JDBC documentation.
Driver Class	Sets the name of the class that implements the JDBC driver. Use this property (along with the dataSourceClass property) only if you do not have a predefined database-type entry in SAP Mobile Server for the kind of SQL database you are connecting to. For example, MySQL database connections require you to use this connection property. To create a connection to a database system, you must use the compatible JDBC driver classes. SAP does not provide these classes; you must obtain them from the database manufacturer.	*`<Class.forName("foo.bar.Driver")>`* Replace <Class.forName("foo.bar.Driver")> with the name of your driver.
Driver Debug	Enables debugging for the driver.	`[true|false]` Set to true to enable debugging, or false to disable.
Driver Debug Settings	Configures debug settings for the driver debugger.	`[default|<setting>]` The default is STATIC:ALL.

Name	Description	Supported values
Initial Pool Size	Sets the initial number of connections in the pool for a datasource. In general, holding a connection causes a less dramatic performance impact than creating a new connection. Keep your pool size large enough for the number of concurrent requests you have; ideally, your connection pool size should ensure that you never run out of available connections. The initialPoolSize value is applied to the next time you start SAP Mobile Server.	`<int>` Replace <int> with an integer to preallocate and open the specified number of connections at start-up. The default is 0. SAP suggests that you start with 0, and create additional connections as necessary. The value you choose allows you to create additional connections before client synchronization requires the server to create them.
Is Download Zipped	Specifies whether the driver file downloaded from jdbcDriverDownloadURL is in `.ZIP` format. This property is ignored if the value of jdbcDriverDownloadURL connection is an empty string.	`[True|False]` The default is false. The file is copied, but not zipped to *`SMP_HOME`*`\lib\jdbc`. Set isDownloadZipped to true to save the file to *`SMP_HOME`*`\lib\jdbc` and unzip the archived copy.
JDBC Driver Download URL	Specifies the URL from which you can download a database driver. Use this property with isDownloadZipped to put the driver in an archive file before the download starts.	`<URL>` Replace <URL> with the URL from which the driver can be downloaded.

Name	Description	Supported values
Language	For those interfaces that support localization, this property specifies the language to use when connecting to your target database. When you specify a value for this property, SAP Mobile Server: • Allocates a CS_LOCALE structure for this connection • Sets the CS_SYB_LANG value to the language you specify • Sets the Microsoft SQL Server CS_LOC_PROP connection property with the new locale information SAP Mobile Server can access Unicode data in an Adaptive Server® 12.5 or later, or in Unicode columns in Adaptive Server 12.5 or later. SAP Mobile Server automatically converts between double-byte character set (DBCS) data and Unicode, provided that the Language and CodeSet parameters are set with DBCS values.	`<language>` Replace <language> with the language being used.
Max Idle Time	Specifies the number of seconds an idle connection remains in the pool before it is dropped.	`<int>` If the value is 0, idle connections remain in the pool until the server shuts down. The default is 60.

Name	Description	Supported values
Max Pool Size	Sets the maximum number of connections allocated to the pool for this datasource. Increase the maxPoolSize property value when you have a large user base. To determine whether a value is high enough, look for ResourceMonitorTimeoutException exceptions in `<hostname>-server.log`. Continue increasing the value, until this exception no longer occurs. To further reduce the likelihood of deadlocks, configure a higher value for maxWaitTime. To control the range of the pool size, use this property with minPoolSize.	`<int>` A value of 0 sets no limit to the maximum connection pool size. The default is 10.
Max Wait Time	Sets the maximum number of seconds to wait for a connection before the request is cancelled.	`<int>` The default is 60.
Max Statements	Specifies the maximum number of JDBC prepared statements that can be cached for each connection by the JDBC driver. The value of this property is specific to each JDBC driver.	`<int>` A value of 0 (default) sets no limit to the maximum statements.
Min Pool Size	Sets the minimum number of connections allocated to the pool for this datasource.	`<int>` A value of 0 (default) sets no limit to the minimum connection pool size.

Name	Description	Supported values
Network Protocol	Sets the protocol used for network communication with the datasource. Use this property (along with the driverClass, and dataSourceClass properties) only if you do not have a predefined database-type entry in SAP Mobile Server for the kind of SQL database you are connecting to. For example, you may be required to use this property for MySQL database connections.	The network protocol is JDBC driver vendor-specific. There are no predefined values. See the driver vendor's JDBC documentation.
Password	Specifies the password for connecting to the database.	`[default|<password>]`
Ping and Set Session Auth	Runs the ping and session-authorization commands in a single command batch; may improve performance. You can only enable the Ping and Set Session Auth property if you have enabled the Set Session Auth property so database work runs under the effective user ID of the client.	`[True|False]` Set to true to enable, or false to disable.
Ping Connections	Pings connections before attempting to reuse them from the connection pool.	`[True|False]` Set to true to enable ping connections, or false to disable.
Ping SQL	Specify the SQL statement to use when testing the database connection with ping.	`[default|<statement>]` Replace <statement> with the SQL statement identifier. The default is "select 1".
Port Number	Sets the server port number where the database server listens for connection requests.	`[default|<port>]` Replace <port> with the TCP/IP port number to use (that is, 1 – 65535). If you set the value as default, the default protocol of the datasource is used.

Name	Description	Supported values
PS Maximum Blob Length	Indicates the maximum number of bytes allowed when updating a BLOB datatype using PreparedStatement.setBytes.	`[default|<int>]` Replace <int> with the number of bytes allowed during an update. The default is 16384.
PS Maximum Clob Length	Indicates the maximum number of characters allowed when updating a CLOB datatype using PreparedStatement.setString.	`[default|<int>]` Replace <int> with the number of bytes allowed during an update. The default is 16384.
Role Name	Sets the database role that the user must have to log in to the database.	`[default|<name>]` If you set this value to default, the default database role name of the datasource is used.
Server Name	Defines the host where the database server is running.	`<name>` Replace <name> with an appropriate name for the server.
Service Name	Defines the service name for the datasource. For SQL Anywhere servers, use this property to specify the database you are attaching to.	`<name>` Replace <name> with an appropriate name for the service.
Set Session Auth	Establishes an effective database identity that matches the current mobile application user. If you use this property, you must also use setSessionAuthSystemID to set the session ID. Alternately you can pingAndSetSessionAuth if you are using this property with pingConnection. The pingAndSetSessionAuth property runs the ping and session-authorization commands in a single command batch, which may improve performance.	`[true|false]` Choose a value of 1 to use an ANSI SQL set session authorization command at the start of each database transaction. Set to 0 to use session-based authorizations.

Name	Description	Supported values
Set Session Auth System ID	If Set Session Authorization is enabled, specifies the database identity to use when the application server accesses the database from a transaction that runs with "system" identity.	`<database identity>` Replace <database identity> with the database identifier.
Start Wait	Sets the wait time (in seconds) before a connection problem is reported. If the start command completes successfully within this time period, no exceptions are reported in the server log. startWait time is used only with the databaseStartCommand property.	`<int>` Replace <int> with the number of seconds SAP Mobile Server waits before reporting an error.
Truncate Nanoseconds	Sets a divisor/multiplier that is used to round the nanoseconds value in a java.sql.Timestamp to a granularity that the DBMS supports.	`[default|<int>]` The default is 10 000 000.
Use Quoted Identifiers	Specifies whether or not SQL identifiers are quoted.	`[True|False]` Set to true to enable use of quoted identifiers, or false to disable.
User	Identifies the user who is connecting to the database.	`[default|<user name>]` Replace <user name> with the database user name.
XA Datasource Class	Specifies the class name or library name used to support two-phase commit transactions, and the name of the XA resource library.	`<class name>` Replace <class name> with the class or library name. • SQL Anywhere database: `com.sybase.jdbc3.jdbc.SybXADataSource` • Oracle database: `oracle.jdbc.xa.client.OracleXADataSource`

See also

- *Managing Connection Pools for SAP Mobile Server Connections* on page 67

SAP Java Connector Properties

Configure SAP Java Connector (JCo) connection properties.

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

This list of properties can be used by all datasource types.

Note: SAP does not document all native endpoint properties. However, you can add native endpoint properties, naming them using this syntax:

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

Standard properties that can be configured within SAP Control Center include:

Table 30. General Connection Parameters for Standard JCo Properties

Name	Description	Supported Values
Enable ABAP Debugging	Enables or disables ABAP debugging. If enabled, the connection is opened in debug mode and you can step through the invoked function module in the debugger. For debugging, an SAP graphical user interface (SAPGUI) must be installed on the same machine the client program is running on. This can be either a normal Windows SAPGUI or a Java GUI on Linux/UNIX systems.	Not supported. Do not set this parameter or leave it set to 0.
Remote GUI	Specifies whether a remote SAP graphical user interface (SAPGUI) should be attached to the connection. Some older BAPIs need an SAPGUI because they try to send screen output to the client while executing.	Not supported. Do not set this parameter or leave it set to 0.

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 RfcGetPartnerSSOTicket() and use it for additional logins to systems supporting the same user base.	Not accessible by the customer. Do not set this parameter or leave it set to 0.
Use X509	SAP Mobile Platform sets this property when a client uses an X509 certificate as the login credential.	If an EIS RFC operation is flagged for SSO (user name and password personalization keys selected in the authentication parameters) then SAP Mobile Platform automatically sets the appropriate properties to use X.509, SSO2, or user name and password SSO credentials. The corresponding properties should not be set by the administrator on the SAP endpoint.
Additional GUI Data	Provides additional data for graphical user interface (GUI) to specify the SAP router connection data for the SAPGUI when it is used with RFC.	Not supported.
GUI Redirect Host	Identifies which host to redirect the remote graphical user interface to.	Not supported.
GUI Redirect Service	Identifies which service to redirect the remote graphical user interface to.	Not supported.
Remote GUI Start Program	Indicates the program ID of the server that starts the remote graphical user interface.	Not supported.

SAP DOE-C Properties

Configure SAP® Data Orchestration Engine Connector (DOE-C) properties. This type of connection is available in the list of connection templates only when you deploy a SAP® Data Orchestration Engine Connector package. No template exists for these types of connections.

Note: If you change the username or password property of a DOE-C connection, you must reopen the same dialog and click `Test Connection` after saving. Otherwise the error state

of this DOE-C package is not set properly, and an error message is displayed. This will not work if you click `Test Connection` before saving the properties.

Name	Description	Supported values
Username	Specifies the SAP user account ID. The SAP user account is used during interaction between the connected SAP system and client for certain administrative activities, such as sending acknowledgment messages during day-to-day operations or "unsubscribe" messages if a subscription for this connection is removed. This account is not used for messages containing business data; those types of messages are always sent within the context of a session authenticated with credentials provided by the mobile client. The technical user name and password or certificateAlias must be set to perform actions on subscriptions. The certificateAlias is mutually exclusive with and overrides the technical user name and password fields if set. The technical user name and password fields can be empty, but only if `certificateAlias` is set.	Valid SAP login name for the DOE host system.
Password	Specifies the password for the SAP user account.	Valid password.
DOE SOAP Timeout	Specifies a timeout window during which unresponsive DOE requests are aborted.	Positive value (in seconds). The default is 420 (7 minutes).

Name	Description	Supported values
DOE Extract Window	Specifies the number of messages allowed in the DOE extract window.	Positive value (in messages). The minimum value is 10. The maximum value is 2000. The default is 50. When the number of messages in the DOE extract window reaches 50% of this value, DOE-C sends a `StatusReqFromClient` message, to advise the SAP DOE system of the client's messaging status and acknowledge the server's state.
Packet Drop Size	Specifies the size, in bytes, of the largest JavaScript Object Notation (JSON) message that the DOE connector processes on behalf of a JSON client. The packet drop threshold size should be carefully chosen, so that it is larger than the largest message sent from the DOE to the client, but smaller than the maximum message size which may be processed by the client.	Positive value (in bytes). The default is 1048576 bytes (1MB). Do not set lower than 4096 bytes; there is no maximum limitation.
Service Address	Specifies the DOE URL.	Valid DOE URL. If you are using DOE-C with SSL: • Modify the port from the standard `http://host:8000` to `https://host:8001/`. • Add the certificate being used as the technical user and DOE-C endpoint security profile certificate to the SAP DOE system's SSL Server certificate list by using the STRUST transaction. See your SAP documentation for details.
Listener URL	Specifies the DOE-C server listener URL.	Valid DOE-C listener URL, for example `http://<sup_hostname>:8000/doe/publish`.

Name	Description	Supported values
SAP Technical User Certificate Alias	Sets the alias for the SAP Mobile Platform keystore entry that contains the X.509 certificate for SAP Mobile Server's SSL peer identity. If you do not set a value, mutual authentication for SSL is not used when connecting to the Web service. If you are using DOE-C with SSO use the "SAP Technical User Certificate Alias" only for configurations which require the technical user to identify itself using an X.509 certificate; it specifies the Certificate Alias to be used as the technical user. This overrides the "Username" and "Password" settings normally used.	Valid certificate alias.
Login Required	Indicates whether authentication credentials are required to login. The default value is true. For upgraded packages, "login-required=false" gets converted to "login-required=true" and a No-Auth security configuration "DOECNoAuth" is assigned to the upgraded package.	A read-only property with a value of true.

Proxy Properties

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

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

Name	Description	Supported Values
Address	Corresponds to the application endpoint provided when registering an application.	Must be a valid application endpoint.
Pool Size	Determines the maximum number of connections allocated to the pool for this datasource.	The default value set for the pool size is 25.
Password	Corresponds to the password of the backend system.	
Allow Anonymous Access	While using REST services, you can enable/disable anonymous user access	The default value is False. To enable anonymous user access, set the value to True.
EnableURLRewrite	This is a custom property which is used to enable/disable URL Rewrite while using REST services.	To enable URL rewrite, set the value to True. To disable URL rewrite, set the value to False.
EnableHttpProxy	This is a custom property which is used to connect to an EIS using HTTP proxy. If you set this property to True, you must configure an HTTP proxy host and port on the server. For more information, see *Configuring SAP Mobile Server to Securely Communicate With an HTTP Proxy in SAP Control Center for SAP Mobile Platform*.	The default value is set to False.

Note:

- In SAP Control Center, when the application endpoint for a registered application is modified under the **Applications** node, you must manually update the **Address** in the proxy properties of the connection pool.
- In SAP Control Center, in addition to the application endpoint, you must register any URL that is required by an application for a proxy service to enable communication with SAP Mobile Server.
- The application end point should be white-listed only once as a proxy connection. The proxy connection name should be same as the application ID, if an application is registered to be used for referencing the EIS service end point.

Web Services Properties

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

Name	Description	Supported Values
Password	Specifies the password for HTTP basic authentication, if applicable.	Password
Address	Specifies a different URL than the port address indicated in the WSDL document at design time.	HTTP URL address of the Web service. A backslash is appended to the address URL, if it does not already exist in the URL you specify.
User	Specifies the user name for HTTP basic authentication, if applicable.	User name
Certificate Alias	Sets the alias for the SAP Mobile Platform keystore entry that contains the X.509 certificate for SAP Mobile 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 SAP Mobile Server certificate keystore.
authentication-Preemptive	When credentials are available and this property is set to the default of false, it allows SAP Mobile 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 SAP Mobile Server to send the authentication credentials in the original SOAP or REST HTTP request message. The message exchange pattern is: request with credentials, a service response.	False (default) True

Name	Description	Supported Values
Socket Timeout	The socket timeout value controls the maximum time, in milliseconds, after a Web service operation (REST or SOAP) is allowed to wait for a response from the remote system; if the EIS does not respond in that time, the operation fails and the SMP thread is unblocked.	Time in milliseconds (default: 6000). Range of [0 – 2147483647], where 0 is interpreted as infinity.
credential.<X>.name	Defines the EIS connection definition to identify the NamedCredential. For more information on token-based SSO using NamedCredential, see *Single Sign-on Using NamedCredential* in *Security*.	• `credential.<X>.name=credential name` • `header:<HTTP header name>` • `cookie: <HTTP cookie name>`
credential.<X>.mapping	Defines how the NamedCredential should be propagated to the EIS. For more information on token-based SSO using NamedCredential, see *Single Sign-on Using NamedCredential*.	`credential.<X>.mapping=credential mapping to header/cookie`
http.header.X.set	Specifies the name of the static header to add to the Web service request (SOAP or REST).	Header name
http.header.X.value	Specifies the content of the header to add to the Web service request (SOAP or REST).	Header value
http.cookie.X.set	Specifies the name of the static cookie to add to the Web service request (SOAP or REST).	Cookie name
http.cookie.X.value	Specifies the content of the cookie to add to the Web service request (SOAP or REST).	Cookie value

Application Connection Properties

(Not applicable to Agentry applications) Set application connection properties through SAP Control Center to create application connections and application connection templates. These settings influence the application connection, including security, datasource connection, user registration, device and application, and so forth.

Application Settings

Application settings display application information, including the Application Identifier, Domain, and Security Configuration

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

  Note: Customization resource bundles are not supported for Hybrid App and Agentry SDK clients.
- **Domain –** the domain selected for the connection template.

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

  Note: Apple and Google do not recommend payload delivery over their systems:
  - Data may not be delivered.

- Data is delivered out of sequence.
- If enabled, the actual payloads must be small.

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

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

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

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

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

Android Push Notification Properties

Android push notification properties allow Android users to install messaging client software on their devices.

Property	Description
Enabled	Enables Google Cloud Messaging (GCM) push notifications to the device if the device is offline. This feature sends a push notification over an IP connection only long enough to complete the Send/Receive data exchange. Android Push notifications overcome issues with always-on connectivity and battery life consumption over wireless networks. Acceptable values: true (enabled) and false (disabled). If this setting is false, all other related settings are ignored.
Registration ID	The registration ID that the device acquires from Google during GCM registration.

Property	Description
Sender ID	GCM sender ID used by SAP Mobile Server to send notifications. Used by the client to register for GCM.

Apple Push Notification Properties

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

- **APNS Device Token –** the Apple push notification service token. An application must register with Apple push notification service for the iOS to receive remote notifications sent by the application's provider. After the device is registered for push properly, this should contain a valid device token. See the iOS developer documentation.
- **Alert Message –** the message that appears on the client device when alerts are enabled. Default: `New items available`.
- **Sounds –** indicates if a sound is a made when a notification is received. The sound files must reside in the main bundle of the client application. Because custom alert sounds are played by the iOS system-sound facility, they must be in one of the supported audio data formats. See the iOS developer documentation.

  Acceptable values: true and false.

  Default: true
- **Badges –** the badge of the application icon.

  Acceptable values: true and false

  Default: true
- **Alerts –** the iOS standard alert. Acceptable values: true and false. Default: true.
- **Enabled –** indicates if push notification using APNs is enabled or not.

  Acceptable values: true and false.

  Default: true

BlackBerry Push Notification Properties

BlackBerry push notification properties enable the server to send notifications to BlackBerry devices using Blackberry Enterprise Server (BES).

Property	Description
Enabled	Enables notifications to the device if the device is offline. This feature sends a push notification over an IP connection only long enough to complete the Send/Receive data exchange. BlackBerry Push notifications overcome issues with always-on connectivity and battery life consumption over wireless networks. Acceptable values: true (enabled) and false (disabled). If this setting is false, all other related settings are ignored. Default: true

Property	Description
BES Push Listener Port	The listener port for BES notifications. The port is discovered and set by the client, and is read-only on the server.
Device PIN	Every Blackberry device has a unique permanent PIN. During initial connection and settings exchange, the device sends this information to the server. SAP Mobile Server uses this PIN to address the device when sending notifications, by sending messages through the BES/MDS using an address such as: Device="Device PIN" + Port="Push Listener port". Default: 0

Capabilities Properties

You cannot update or modify the capabilities properties. These properties are set by the client application and determine what properties can be set for the application connection.

- **Application Supports Client Callable Components –** Supports tracing, package users, cloning, and reregistering for application connections.
- **Application Supports Hybrid App –** Supports assigning Hybrid Apps to an application connection.
- **Application Supports Password Policy –** Supports updating the password policy for an application connection.

Connection Properties

Connection properties define the connection information for a client application so it can locate the appropriate SAP Mobile Server synchronization service.

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

- **Activation Code –** (not applicable to replication clients) the original code sent to the user in the activation e-mail. Can contain only letters A – Z (uppercase or lowercase), numbers 0 – 9, or a combination of both. Acceptable range: 1 to 10 characters.
- **Farm ID –** a string associated with the Relay Server farm ID. Can contain only letters A – Z (uppercase or lowercase), numbers 0 – 9, or a combination of both. Default: 0.
- **Server Name –** the DNS name or IP address of the SAP Mobile Server, such as "myserver.mycompany.com". If using Relay Server, the server name is the IP address or fully qualified name of the Relay Server host.
- **Server Port –** the port used for messaging connections between the device and SAP Mobile Server. If using Relay Server, this is the Relay Server port. Default: 5011.
- **Synchronization Server Host –** the server host name used for synchronization.
- **Synchronization Server Port –** the port used for synchronization.

- **Synchronization Server Protocol –** the synchronization protocol - HTTP or HTTPS.
- **Synchronization Server Stream Parameters –** the synchronization server stream parameters that are used to explicitly set client-specific values. After the client application successfully registers with the SAP Mobile Server, it receives the trusted certificate configured in the Server configuration (either the Secure Sync Port Public Certificate or Trusted Relay Server Certificate). If you are using Relay Server, ensure the Trusted Relay Server Certificate property is configured to point to a file that has the server's public certificate.

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

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

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

- **Use HTTPS –** forces clients to use HTTPS protocol for the connection.

Custom Settings

Define one of four available custom strings that are retained during reregistration and cloning.

Change the property name and value according to the custom setting you require. The custom settings can be of variable length, with no practical limit imposed on the values. You can use these properties to either manually control or automate how Hybrid App-related messages are processed:

- Manual control – an administrator can store an employee title in one of the custom fields. This allows employees of a specific title to respond to a particular message.
- Automated – a developer stores the primary key of a back-end database using a custom setting. This key allows the database to process messages based on messaging device ID.

Device Advanced Properties

Advanced properties set specific behavior for messaging devices.

- **Relay Server URL Prefix –** the URL prefix to be used when the device client is connecting through Relay Server. The prefix you set depends on whether Relay Server is installed on IIS or Apache. For IIS, this path is relative. Acceptable values include:

  - For IIS – use`/ias_relay_server/client/rs_client.dll`.

- For Apache – use `/cli/iasrelayserver`.

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

- **Allow Roaming –** the device is allowed to connect to server while roaming. Acceptable values: true and false. Default: true.
- **Debug Trace Size –** the size of the trace log on the device (in KB). Acceptable values: 50 to 10,000. Default: 50.
- **Debug Trace Level –** the amount of detail to record to the device log. Acceptable values: 1 to 5, where 5 has the most level of detail and 1 the least. Default: 1.
  - 1: Basic information, including errors
  - 2: Some additional details beyond basic
  - 3: Medium amount of information logged
  - 4: Maximum tracing of debugging and timing information
  - 5: Maximum tracing of debugging and timing information (currently same as level 4)
- **Device Log Items –** the number of items persisted in the device status log. Acceptable values: 5 to 100. Default: 50.
- **Keep Alive (sec) –** the Keep Alive frequency used to maintain the wireless connection, in seconds. Acceptable values: 30 to 1800. Default: 240.

Device Info Properties

Information properties display details that identify the mobile device, including International Mobile Subscriber identity (IMSI), phone number, device subtype, and device model.

- **IMSI –** the International Mobile Subscriber identity, which is a unique number associated with all Global System for Mobile communication (GSM) and Universal Mobile Telecommunications System (UMTS) network mobile phone users. To locate the IMSI, check the value on the SIM inside the phone.
- **Phone Number –** the phone number associated with the registered mobile device.
- **Device Subtype –** the device subtype of the messaging device. For example, if the device model is a BlackBerry, the subtype is the form factor (for example, BlackBerry Bold).
- **Model –** the manufacturer of the registered mobile device.

Password Policy Properties

Create a password policy for device application logins. Only passwords that meet the criteria of the policy can be used to access the sensitive artifacts secured inside a device's data vault.

You can create a password policy as part of an application connection template. Ensure your developers add enforcement code to the application's data vault.

- **Enabled –** Set this value to `True` to enable a password policy for device applications. By default, this property is set to True.
- **Default Password Allowed –** Set this value to `True` to allow default passwords. If a default password is allowed in the policy, developers can create the vault using with a default password, by specifying null for both the salt and password arguments. By default, this value is set to False
- **Expiration Days –** Sets the number of days the existing password can be used before it must be changed by the user. By default, this value is set to 0, or to never expire.
- **Has Digits | Lower | Special | Upper –** Determines what combination of characters must be used to create a password stringency requirements. The more complex the password, the more secure it is deemed to be. Set the value to `True` to enable one of these password stringency options. By default they are set to false.

Note: Any non-alphanumeric characters are considered to be special characters.

- **Lock Timeout –** Determines how long a successfully unlocked data vault will remain open. When the timeout expires, the vault is locked, and the user must re-enter the vault password to resume using the application. Use this property in conjunction with the Retry Limit.
- **Minimum Length –** Sets how long the password chosen by the user must be. By default, this value is set to 8.
- **Minimum Unique Characters –** Determines how many unique characters must be used in the password. By default this property is set to 0. For example, if set that the password has a minimum length of 8 characters, and the number of unique characters is also 8, then no duplicate characters can be used. In this instance a password of `Sm00the!` would fail, because two zeros were used. However, `Smo0the!` would pass because the duplication has been removed.
- **Retry Limit –** Sets the number of times an incorrect password can be retried before the data vault is deleted. A deleted vault means that the database encryption key is lost, and all data in the application is rendered irretrievable. As a result the application becomes unusable. By default this value is set to 20.

Proxy Properties

(Applies only to Online Data Proxy) Proxy properties display details that identify the application and push endpoints.

- **Application Endpoint –** the URL pointing to the EIS.
- **Push Endpoint –** the SAP SAP Mobile Platform URL used by EIS to forward notifications.

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

Security Settings

Security settings display the device security configuration.

- E2E Encryption Enabled – indicate whether end-to-end encryption is enabled or not: true indicates encryption is enabled; false indicates encryption is disabled.
- E2E Encryption Type – use RSA as the asymmetric cipher used for key exchange for end-to-end encryption.
- TLS Type – use RSA as the TLS type for device to SAP Mobile Server communication.

Note: These settings are visible, but not in use by client (replication native application) at this time.

User Registration

User registration specifies details of the activation code that is sent to a user to manually activate an application on the device.

- Activation Code Expiration (Hours) – indicates how long an activation code is valid. The default is 72 hours.
- Activation Code Length – indicates the length of the activation code, as in number of alphanumeric characters. The default is 3.

Command Line Utilities

Invoke command line utilities directly from the operating system.

SAP Mobile Platform includes a set of command line utilities for carrying out routine administrative tasks, such as deploying a package or maintaining a cache database (CDB). You can also include these commands in batch files for repeated use.

To launch a utility:

- Double-click its icon, if it has one.
- If it does not have an icon in the program group, enter the utility program command at the Windows command prompt.

Relay Server Utilities

Use relay server utilities to administer the relay server: rshost and regRelayServer.

Launch these utilities from the command line; they are not available from any other administration tool.

Relay Server Host (rshost) Utility

State Manager (rshost) maintains state information across Relay Server client requests and RSOE sessions, and manages the Relay Server log file. It also provides a command line utility

on the Relay Server host, to update the Relay Server configuration and archive the Relay Server log.

Syntax

Variable declaration:

```
rshost [option]+
```

Parameters

- ***<option>* –**

Option	Description
`-f <filename>`	Relay Server configuration file
`-o <filename>`	Relay Server log file
`-oq`	Prevents popup window on startup error
`-q`	Run in minimized window
`-qc`	Close window on completion
`-u`	Update Relay Server configuration
`-ua`	Archive Relay Server log file to `<yymmdd><nn>.log` and truncate

Usage

For more information, see "Relay Server State Manager command line syntax" in *SQL Anywhere documentation*.

Register Relay Server (regRelayServer) Utility

Use `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\bin\regRelayServer.bat` to perform multiple relay server and RSOE administrative actions that can also be performed in SAP Control Center.

Syntax

```
regRelayServer [config | export | exportRSconfig | migrate|
quickconfig | remove | start | stop]
```

Parameters

- **config –** Configure relay server or an RSOE for an SAP Mobile Server node with a named XML file as input. Supported options include:

  - **-f <*filename*> –** The input file and path.
- **export –** Export the existing RSOE configuration into a file. Supported options include:
  - **-o <*filename*> –** The output file and path.
- **exportRSconfig –** Export the actual relay server configuration properties file, which could be used by the SAP Mobile Platform administrator to configure SAP Mobile Server farms, server nodes, for remaining relay servers that require configuration or updates. Supported options include:
  - **-o <*filename*> –** The output file and path.
  - **-h <*name*> –** (Required) The name of the host.
  - **-p <*port*> –** (Required) The port number used by the relay server.
- **migrate –** Migrate the configuration values defined in the 1.5.5 version of *<UnwiredPlatform_InstallDir>*`\Servers\UnwiredServer\config\relayserver.properties` file and port them to the 1.6 `rsconfig.xml` file. Be aware that:
  - The `relayserver.properties` file must exist and must use supported values for the relay server and RSOE. Ensure that no changes to the file occur while the file is being processed by this command.
  - This command is only valid until `relayserver.properties` has been migrated for a specific relay server host.
  - Once you migrate `relayserver.properties`, do not update it.
- **quickconfig –** Rapidly create a relay server configuration, collecting only connection property values and use system-generate defaults for the remaining properties required. RSOEs deployed with this method require manual property edits before they can be started. Export the configuration and update the properties in the file. You can then re-apply the configuration with the configure command.
- **remove –** Deletes one or all RSOE process for the SAP Mobile Server node. However, the configuration file is not deleted, because other RSOEs on the server node may require it. Supported options include:
  - **-f <*name*> –** The relay server farm name.
  - **-h <*name*> –** The name of the relay server host.
  - **-p <*port*> –** (Required) The port number used by the relay server.

  If you do not specify -h, -p, and -f options, all RSOEs for the SAP Mobile Server node are removed. Otherwise, only those identified are removed.
- **start –** Start one or all RSOE processes for the SAP Mobile Server node. Supported options include:
  - **-f <*name*> –** The relay server farm name.
  - **-h <*name*> –** The name of the relay server host.
  - **-p <*port*> –** (Required) The port number used by the relay server.

If you do not specify -h, -p, and -f options, all RSOEs for the SAP Mobile Server node start. Otherwise, only those identified are started.

- **stop –** Stop one or all RSOE processes for the SAP Mobile Server node. Supported options include:
  - **-f <*name*> –** The relay server farm name.
  - **-h <*name*> –** The name of the SAP Mobile Server host.
  - **-p <*port*> –** (Required) The port number used by the SAP Mobile Server.

  If you do not specify -h, -p, and -f options, all RSOEs for the SAP Mobile Server node stop. Otherwise, only those identified are stopped.

Examples

- **Export properties then configure a new relay server –** In this example, an administrator uses this utility multiple times but on different host computers:
  1. The administrator exports the file for a relay server deployed in a test environment. This configuration is stable and the administrator now wishes to propagate these properties to all new relay servers in a production environment. The administrator runs this command with the on the test host computer:

     ```
     regrelayserver.bat exportRSconfig -o <path>\rsconfig.xml
     ```
  2. The administrator transfers the file to the new host. The command used to apply the file on the new host is:

     ```
     regrelayserver.bat config -f <path>\rsconfig.xml
     ```
- **Registering a new RSOE and configuring it with rsoe.config.template.xml –** In this example an administrator has copied and modified *SMP_HOME*`\Servers\UnwiredServer\config\rsoeConfig.template.xml`.. Because the XML elements do not include an `ID` attribute value, this utility will treat the RSOE as newly deployed, and register it in the cluster database in addition to configuring it. The administrator uses this command:

  ```
  regrelayserver.bat config -f <path>\rsoe.config.template.xml
  ```
- **Start all RSOEs on the SAP Mobile Server node –** Once RSOEs are configured, rather than restarting the host computer, the administrator starts all newly configured RSOEs on the current machine with this command:

  ```
  regrelayserver.bat start
  ```
- **Stop only RSOEs of a named relay server farm –** An administrator needs to make a change to one of the RSOEs recently started, and uses this command:

  ```
  regrelayserver.bat stop -f myhost.MBS.farm -h myUServerhost -p
  5011
  ```

Usage

Ensure the SAP Mobile Server node cluster database is running before you use this utility.

RSOE Service (rsoeservice) Utility

Installs, removes, starts, and stops the relay server outbound enabler (RSOE) as a Windows service from the command line. This utility is located in *SMP_HOME*\Servers\UnwiredServer\bin.

Use this utility to manage RSOE services. To apply the changes in relayserver.properties, use **regRelayServer.bat**.

Syntax

```
rsoeservice [install auto | install manual | remove | start | stop]
```

Parameters

- **install auto –** installs RSOE as a windows service in auto-start mode.
- **install manual –** installs RSOE as a windows service in manual start mode.
- **remove –** uninstalls the RSOE service.
- **start –** starts the RSOE service.
- **stop –** stops the RSOE service.

Examples

- **Basic Example –** This command installs RSOE as an auto-start Windows service:

  ```
  rsoeservice install auto
  ```

Certificate and Key Management Utilities

Use the certificate management utilities to encrypt SAP Mobile 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 *SMP_HOME*\Servers\SQLAnywhere*XX*\BIN*XX*.

You may choose to purchase certificates from a third party. These certificate authorities (CAs) provide their own tools for creating certificates. You can use **createcert** to create certificates for development and testing; you can also use it for production certificates.

Syntax

```
createcert [options]
```

Parameters

- **[*options*] –** these options are available through the **createcert** utility:

Option	Description
`-r`	Creates a PKCS #10 certificate request. **createcert** does not prompt for a signer or any other information used to sign a certificate.
`-s <filename>`	Signs the PKCS #10 certificate request that is in the specified file. The request can be DER or PEM encoded. **createcert** does not prompt for key generation or subject information.

Note: To create a signed certificate, use **createcert** without options. To break the process into two separate steps, for example so one person creates a request and another person signs it, the first person can run **createcert** with `-r` to create a request and the second person can sign the request by running **createcert** with `-s`.

When you run **createcert**, you are asked for all or some of this information, depending on whether you specified `-r`, `-s`, or neither.

- `Choose encryption type` – select RSA.
- `Enter RSA key length (512-16384)` – this prompt appears only if you chose RSA encryption. Specify a length between 512 bits and 16384 bits.
- Subject information – enter this information, which identifies the entity:
  - `Country Code`
  - `State/Province`
  - `Locality`
  - `Organization`
  - `Organizational Unit`
  - `Common Name`
- (Optional) `Enter file path of signer's certificate` – supply a location and file name for the signer's certificate. If you supply this information, the generated certificate is a signed certificate. If you do not supply this information, the generated certificate is a self-signed root certificate.
- `Enter file path of signer's private key` – supply a location and file name to save the private key associated with the certificate request. This prompt appears only if you supplied a file in the previous prompt.
- `Enter password for signer's private key` – supply the password that was used to encrypt the signer's private key. Supply this password only if the private key was encrypted.
- (Optional) `Serial number` – supply a serial number. The serial number must be a hexadecimal string of 40 digits or less. This number must be unique among all

certificates signed by the current signer. If you do not supply a serial number, **createcert** generates a GUID as the serial number.

- `Certificate will be valid for how many years (1-100)` – specify the number of years for which the certificate is valid. After this period, the certificate expires, along with all certificates it signs.
- `Certificate Authority (y)es or (n)o` – indicate whether this certificate can be used to sign other certificates. The default value is no.
- `Key usage` – supply a comma-separated list of numbers that indicate the ways in which the certificate's private key can be used. The default, which depends on whether the certificate is a certificate authority, should be acceptable for most situations.
- `File path to save request` – this prompt appears only if you specify the -r option. Supply a location and file name for the PCKS #10 certificate request. Supply a location and file name in which to save the certificate. The certificate is not saved unless you specify a location and file name.
- `Enter file path to save private key` – supply a location and file name in which to save the private key. Enter a password to protect private key. Optionally, supply a password with which to encrypt the private key. If you do not supply a password, the private key is not encrypted. This prompt appears only if you supplied a file in the previous prompt.
- `Enter file path to save identity` – supply a location and file name in which to save the identity. The identity file is a concatenation of the certificate, signer, and private key. This is the file that you supply to the server at start-up. If the private key was not saved, **createcert** prompts for a password to save the private key. Otherwise, it uses the password provided earlier. The identity is not saved unless you provide a file name. If you do not save the identity file, you can manually concatenate the certificate, signer, and private key files into an identity file.

Examples

- **Example 1: –** creates a self-signed certificate. No file name is provided for the signer's certificate, which makes it a self-signed root certificate.

```
SMP_HOME\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 SAP Mobile Server end-to-end encryption. This utility is located in *SMP_HOME*\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
```

Key Tool (keytool) Utility

keytool is a JDK utility used to manage a keystore (database) of private keys and associated X.509 certificates, as well as certificates from trusted entities. **keytool** is in *SMP_HOME* `\Servers\SQLANywhere11\Sun\JRE160_x86\bin`.

keytool enables users to create and manage their own public/private key pairs and associated certificates for use in self-authentication or data integrity and authentication services, using digital signatures. It also allows users to cache the public keys (in the form of certificates) of their communicating peers.

Syntax

```
keytool -list | -printcert | -import | -export| -delete | -selfcert |
-certreq | -genkey [options]
```

Parameters

- **-list –** displays the contents of a keystore or keystore entry.
- **-printcert –** displays the contents of a certificate stored in a file. Check this information before importing a certificate as a trusted certificate. Make sure certificate prints as expected.
- **-import –** imports:
  - a certificate or certificate chain to the list of trusted certificates, or,
  - a certificate reply received from a certificate authority (CA) as the result of submitting a certificate signing request (CSR).

  The value of the `-alias` option indicates the type of import you are performing. If the alias exists in the database, then it is assumed you want to import a certificate reply. **keytool** checks whether the public key in the certificate reply matches the public key stored with the alias, and exits if they do not match. If the alias identifies the other type of keystore

entry, the certificate is not imported. If the alias does not exist, it is created, and associated with the imported certificate.

- **-export –** exports a certificate to a file.
- **-delete –** deletes a certificate from the list of trusted certificates.
- **-selfcert –** generates a self-signed certificate. The generated certificate is stored as a single-element certificate chain in the keystore entry identified by the specified alias, where it replaces the existing certificate chain.
- **-certreq –** generates a certificate signing request (CSR), using the PKCS #10 format. A CSR is intended to be sent to a CA, which authenticates the certificate requestor and returns a certificate or certificate chain, used to replace the existing certificate chain in the keystore.
- **-genkey –** generates a key pair (a public key and associated private key). Wraps the public key into an X.509 v1 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are stored in a new keystore entry identified by <*alias*>.

-genkey Option	Description
`-keystore <keyStoreLocation>`	Name and location of the persistent keystore file for the keystore managed by **keytool**. If you specify, in the `-keystore` option, a keystore that does not exist, a keystore is created. If you do not specify a `-keystore` option, the default keystore is a file named `.keystore` in your home directory. If that file does not exist, it is created.
`-storepass <password>`	The password protecting keystore integrity. The password must be at least 6 characters long and provided to all commands that access the keystore contents. For such commands, if a `-storepass` option is not provided at the command line, the user is prompted for it.
`-file <certificateFile>`	The certificate file location.
`-noprompt`	During import, removes interaction with the user.
`-trustcacerts`	When importing a certificate reply, the certificate reply is validated using trusted certificates from the keystore, and using the certificates configured in the `cacerts` keystore file. This file resides in the JDK security properties directory, `java.home\lib\security`, where java.home is the runtime environment's directory. The `cacerts` file represents a system-wide keystore with CA certificates. System administrators can configure and manage that file using **keytool**, specifying "jks" as the keystore type.
`-alias <alias>`	The logical name for the certificate you are using.

-genkey Option	Description
`-keypass <pass-word>`	The password that protects the private key of the key pair. Press Enter at the prompt to set the key password to the password associated with the keystore. `keypass` must be at least 6 characters long.

Examples

- **Example 1: Display the contents of the keystore –** `keytool -list -keystore <filePath>\keystore.jks -storepass <storepass>`
- **Example 2: Import a certificate reply from a CA –** `keytool -import -file <certificate file> -keystore <filePath>\keystore.jks -storepass <storepass> -noprompt -trustcacerts -alias <alias>`
- **Example 3: Delete a certificate –** `keytool -delete -alias <alias> -keystore <filePath>\keystore.jks -storepass <storepass>`
- **Example 4: Generate a key pair –** `keytool -genkey -keystore <filePath>\keystore.jks`

  The certificate request must be signed by a CA or self-signed by using the `-selfcert` **keytool** option.

SAP Mobile Server Runtime Utilities

SAP Mobile Server runtime supports many utilities used to manage the server environment and its artifacts.

Launch these utilities from the command line, or from SAP Control Center.

License Upgrade (license) Utility

Upgrades the license in a served model when you purchase more licenses from SAP. In an unserved model, you simply replace the license file. This utility is located in *SMP_HOME* `\Servers\UnwiredServer\bin`.

The `license.bat` utility upgrades only SAP Mobile Platform licenses. You cannot use it to upgrade Afaria licenses, which can be upgraded only with the Afaria Administrator.

Syntax

```
license.bat <PE> <LT> [LicenseNumber]
```

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

Parameters

- **PE –** use the corresponding abbreviation that matches the product edition in your license. Valid product editions include:
  - EE for Enterprise Edition
  - ED for Enterprise Developer Edition
  - PD for Personal Developer Edition
- **LT –** use the corresponding abbreviation that matches the license type in your license. Valid license types include:
  - AC for Application deployment CPU license
  - AS for Application deployment seat license
  - CP for CPU license (client-server licenses)
  - ST for Seat license
  - OT for Other license
  - SS for Standalone seat license
  - DT for Development and test license

Examples

- **Update Client and Server Licenses –** To update the client and server license number from the current value to 1200 in an enterprise edition of SAP Mobile Platform, run:

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

Usage

Depending on the product edition you have, the license type varies according to these guidelines:

- If you entered EE for product edition, the license type can be AC, AS, CP, or ST.
- If you entered PD for product edition, the license type can be SS.
- If you entered ED for product edition, the license type can be DT.

Synchronization Monitor (mlmon) Utility

Monitors the progress of replication synchronization and checks for potential data contention or bottlenecks. This utility is located in *SMP_HOME* `\Servers\SQLAnywhere`*XX* `\BIN32`.

Prerequisites: To use this utility, ensure you are using the correct version of the JDK. For details, see *SAP Mobile Platform Runtime Requirements* in *Supported Hardware and Software*. If you choose to use an existing JDK option during installation, also ensure that you have set the `JAVA_HOME` environment variable to the installation location of this JDK version. Otherwise, you must use a text editor to modify the existing JDK path in the batch file itself.

Syntax

```
mlmon [connect-options|inputfile.{mlm|csv}]
```

When you execute this command, you are prompted to provide:

- Valid administrator credentials
- Replication protocol (default: HTTP)
- Replication port (default: 2480)

Parameters

- **connect-options –** allows you to connect to the SAP Mobile Server on startup. A monitor connection starts like a synchronization connection to the SAP Mobile Server. Allowed values are:

Option	Description
`-u ml_username`	Required to connect to the SAP Mobile Server.
`-p password`	Required to connect to the SAP Mobile Server.

`-x [tcpip | tls | http | https] [(keyword=value;...)]`	Required to connect to the SAP Mobile Server. The keyword=value pairs can be:
	• Host – the network name or IP address of the computer where the SAP Mobile Server is running. By default, it is the computer where the monitor is running.
	• Protocol – should be set to the same network protocol and port as the SAP Mobile Server is using for synchronization requests.
	• Additional Network Parameters – optional parameters, including:
	  • `buffer_size=number`
	  • `client_port=nnnn`
	  • `client_port=nnnn-mmmmm`
	  • `persistent=[0|1]` (HTTP and HTTPS only)
	  • `proxy_host=proxy_hostname` (HTTP and HTTPS only)
	  • `proxy_port=proxy_portnumber` (HTTP and HTTPS only)
	  • `url_suffix=suffix` (HTTP and HTTPS only)
	  • `version=HTTP-version-number` (HTTP and HTTPS only)

Option	Description
`-o outputfile.{mlm|csv}`	Closes the monitor at the end of the connection and saves the session in the specified file.

- **inputfile.{mlm|csv} –** prompts the monitor to open the specified file.

Package Administration (supadmin.bat) Utility

Deploy, delete, import, and export application packages using the administration with supadmin.bat and its supported commands. You can execute these commands from the administration command line utility console or from the command line.

To issue commands using the interactive administration command line utility console, open *SMP_HOME*\Servers\UnwiredServer\bin\supadmin.bat. You must enter your host name, port, and administrator credentials before entering a command. Otherwise, the console prompts you for:

- Host – the host name for the SAP Mobile Server. The default value is localhost.
- Port – the port used for the SAP Mobile Server. The default value is 2000.
- Login ID – the administrator login ID to perform authentication with. The default value is supAdmin.
- Password – the administrator password to perform authentication with. The default value is s3pAdmin.

To issue the utility at the command line use:

```
supadmin.bat -host host name -port port -u username -pw password
Deploy|Import|Export|Delete commandOptions
```

Deploy Command

Deploy a package to SAP Mobile Server with the administration client APIs.

Syntax

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

Parameters

- **deploy-options –** uses these option to deploy and configure the package:

Option	Description
-d *domain*	Specifies the domain to which the package belongs.
-dcf *descriptorFile*	(Optional) Specifies the descriptor file for the package.

Option	Description
-dm *deploymentMode*	The deployment mode determines how the deployment process handles the objects in a deployment unit and package. Which value you choose depends on whether or not a package of the same name already exists on SAP Mobile Server. Allowed values are: • `UPDATE` – updates the target package with updated objects. After deployment, objects in the server's package with the same name as those being deployed are updated. By default, deploymentMode is UPDATE. • `VERIFY` – do not deploy package. Only return errors, if any. Used to determine the results of the `UPDATE` deploy mode. If the deployment mode is specified both in the descriptor file and the command-line, the command-line deploymentMode option override the deployment mode specified in the descriptor file.
-file *deploymentUnitFileName*	Defines the file name of the deployment unit. For example, `MyDeployUnit.xml`.
-rm *roleMapping*	Defines the role mapping for the package. Accepted values are: • `role1=AUTO` – the logical role defined in the package. • `role2=SUPAdmin,SUPUser` – the physical roles defined in the security provider. The role mapping `-rm role1,role2` maps the logical roles of the package to the physical roles in the server.
-sc *securityConfig*	Defines the security configuration for the package. Select an existing security configuration from the domain to which the package will be deployed.
-sl	Enables silent mode, which disables all user interactive questions during package deployment. During silent mode, the default values for each option are used. This mode is mainly used when writing a batch executing file. For example, the command line `deploy -file deployunit.xml -sl` deploys the package to the default domain with a deploy mode of `UPDATE` and a sync mode of `REPLICATION`, without asking for user confirmation.

Examples

- **Basic Example –** This command updates an existing deployment unit called `samples/uep/deployment/customer_list_unit.xml`. The batch file uses default values for all other command line options:

```
supadmin -host test01.sap.com -port 2000 -u supAdminID -pw supPwd
deploy -d default -dm update -sc admin -file samples/uep/
deployment/customer_list_unit.xml
-rm "testrole=SUP Admin;testrole1=SUP User,SUP User2"
```

Import and Export Commands

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

Syntax

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

To import an MBO package:

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

To export an MBO package:

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

To import a Hybrid App package:

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

To export a Hybrid App package:

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

To import an application:

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

To export an application:

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

Examples

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

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

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

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

Usage

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

A successful package import requires that:

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

A successful application import requires that::

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

A Hybrid App import has no dependencies.

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

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

Delete Command

Delete a package from SAP Mobile Server.

Syntax

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

Parameters

- **delete-options –** use these options to delete the package:

Option	Description
-d *domain*	Specifies the domain to which the package belongs.
-pkg *packageName*	Specifies the name of the package to delete.

Examples

- – This command deletes samplePkg from SAP Mobile Server:

```
supadmin.bat -host test01.sap.com -port 2000 -u supAdmin -pw PWD
delete   -d default -pkg samplePkg
```

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

Creates or removes the SAP Mobile Platform SampleDB Windows service for the sampledb server. Used on an SAP Mobile Server that was not installed with a development license.

Syntax

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

Parameters

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

Update Properties (updateprops.bat) Utility

Performs multiple functions, including registering or removing a participating node for a cluster, or update a specific server property.

Note: Unless documented otherwise, use SAP Control Center to change most properties in the runtime to avoid unnecessary complication.

Syntax

```
updateprops.bat [-u username] [-p password] [-d dsn]
[-cn clusterName] [-nv "<propertyName=NewValue>"] [-v]
```

Parameters

- **-u *username* –** the platform administrator username.
- **-p *password* –** the platform administrator password.
- **-d *dsn* –** the data source name (DSN) of the cluster database.
- **-cn *clusterName* –** the name that identifies the SAP Mobile Platform Cluster
- **-nv "<*propertyName=NewValue*>" –** one or more platform property values that requires change. Multiple values can be defined; however, they must be separated by the pound symbol (#). For example:

  ```
  –nv
  "ml.threadcount=10#sup.admin.port=2005#sup.sync.port=2490"
  ```
- **-v –** use verbose output in the command window.

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

Examples

- **Changing a cdb threadcount property –** Update the ml.threadcount property of the production environment cache database to `20` by running:

  ```
  updateProps.bat -nv "ml.threadcount=20"
  ```

  This is only recommended for deployment editions of SAP Mobile Platform.

Usage

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

See also

- *Changing Database Ports for SQLAnywhere Databases* on page 63
- *Changing SQLAnywhere Database Server Startup Options* on page 64

Diagnostic Tool Command Line Utility (diagtool.exe) Reference

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

Default file location: The **diagtool.exe** tool is located in the SAP Mobile Platform installation folder at *SMP_HOME*`\Servers\UnwiredServer\diagtool`.

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

Syntax

```
diagtool verify options
```

Parameters

This command line utility can be used to validate different aspects of SAP Mobile Platform. Parameters are organized by supported use cases:

Table 31. Parameters for Command Support

Parameter	Description
verify	Verify specific configuration

Table 32. Parameters for Supported Sub-commands

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

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

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

Table 34. Parameters for Messaging Verification

Parameter	Description
-push	Validate messaging connection

Table 35. Parameters for Security Configuration Verification

Parameter	Description
-security	Verify security configuration
-sc	Security configuration

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

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

Table 37. Parameters for Diagtool Verbosity Option

Parameter	Description
-verbose	(Optional) Output detail trace information

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

Parameter	Description
-e2ee	Validate end-to-end encryption

Table 39. Parameters for Replication Synchronization Verification

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

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

Table 40. Parameters for Automatic Registration Verification

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

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

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

Examples

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

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

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

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

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

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

  For more information, see *Syntax and Scenarios*.

Diagnostic Tool Command Line Utility

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

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

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

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

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

See also

- *Syntax and Scenarios* on page 289

Syntax and Scenarios

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

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

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

- **Scenario: Verify connection to SAP Mobile Server messaging port** - Verify that the messaging client is connecting to SAP Mobile Server directly.

  ```
  diagtool verify -push -h host -P port -u username -p password
  ```

- **Scenario: Verify the connection to SAP Mobile Server messaging port via a Relay Server** - Verify the messaging client connection to SAP Mobile Server via Relay Server.

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

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

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

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

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

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

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

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

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

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

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

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

See also

- *Diagnostic Tool Command Line Utility* on page 289

DOE-C Utilities

To perform tasks needed when working with SAP DOE Connector.

esdma-converter Command

Converts an SAP ESDMA bundle resource metadata file to an SAP Mobile Platform package.

The **esdma-converter** executable file is located in the *SMP_HOME*`\MobileSDK\ObjectAPI\Utils\bin` directory.

Syntax

```
esdma-converter esdma-bundle-dir [-afx afx_file]
[-dsd output_file] [-esdma bundle_metadata_file] [-help]
```

Parameters

- ***esdma-bundle-dir* –** the source directory for the ESDMA resource metadata file to be converted.
- **-afx –** the AFX (application from XML) output document file. The default is *esdma-bundle-dir*/META-INF/afx-esdma.xml.
- **-dsd –** the DOE-C output document name. The default is *esdma-bundle-dir*/META-INF/ds-doe.xml.
- **-esdma –** the source ESDMA bundle resource metadata file. The default is *esdma-bundle-dir*/Resources/AnnotatedMeta.xml, or *esdma-bundle-dir*/Resources/Meta.xml if the former is not found.
- **-help –** gets help on this command.

Code Generation Utility Command Line Reference

Generates platform-specific code for a message-based application from an ESDMA bundle.

Before using the Code Generation Utility:

- Copy your ESDMA .zip file into an ESDMA directory (*<ESDMA_dir>*) and extract its contents into that directory.
- Verify that the following files are present in the META-INF directory under your *<ESDMA_dir>* directory.
  - sup-db.xml
  - afx-esdma.xml
  - ds-doe.xml

Note: *<ESDMA_dir>*\META-INF\sup-db.xml is also referred to as the AFX document.

---

codegen Command

Command syntax and parameter descriptions.

Run codegen.bat to execute the Code Generation Utility. The codegen.bat file is located in *SMP_HOME*\MobileSDK\ObjectAPI\Utils\bin.

Syntax

```
codegen -android|-cs|-oc -client -doe -sqlite|-ulj
[-output <output_dir>] [-doc] <ESDMA_dir>\META-INF\sup-db.xml
```

All parameters not enclosed by "[...]" are required for use with DOE-C.

Parameters

- **-cs –** for Windows and Windows Mobile, generates C# source files.
- **-android –** for Android, generates Java source files.
- **-oc –** for iPhone, generates Object C source files.
- **-client –** generates client side source code.
- **-doe –** generates code for a DOE-based synchronization package.
- **-sqlite –** for Windows and Windows Mobile, and iPhone, generates code for database type SQLite.
- **-ulj –** for BlackBerry, generates code for database type UltraLiteJ.
- **-output –** specifies the output directory where generated code is placed. Defaults to *SMP_HOME*`\MobileSDK\ObjectAPI\Utils\genfiles`.
- **-doc –** generates documentation for the generated code.

SAP DOE Connector Command Line Utility

A text-based console that allows you to manage ESDMA packages and subscriptions to those packages without going through SAP Control Center.

Work interactively in the Command Line Utility console until you become familiar with the command options. The console prompts you for all required parameters and lets you choose values from a list, when there is a fixed list of valid values; for example, for domain name.

Write batch files to silently execute any sequence of commands.

You must use the Command Line Utility's **deploy** command to deploy an ESDMA package. The functionality of all the other commands is available through SAP Control Center. See *SAP Control Center for SAP Mobile Platform > Deploy > DOE-C Packages*.

Note: Before you attempt to deploy a package, set up the security configuration in SAP Control Center for that package to use. See *SAP Control Center for SAP Mobile Platform > Administer > Security Configurations*.

Table 41. Command Line Utility Notation Conventions

Symbols	Meaning	Example
[...]	Optional – parameters not enclosed by this symbol are required	`[-h|--help]` Getting help is an option on every command.
|	Alternatives – use one or the other, not both	`[-h|--help]` You can enter the help parameter as `-h` or as `--help`.

Symbols	Meaning	Example
{...}	Grouping – alternatives symbol applies to all parameters enclosed	**{**`-u|--technicalUser` *`SAPUserAccount`* `-pw|--password` *`SAPUserPassword`***}** **|** **{**`-ca|-certAlias` *`certificateAlias`***}** You must enter either the technical user ID and password, or the certificate alias.

Starting the Command Line Utility Console

Before you can use the DOE-C Command Line Utility interactively, you must start the console.

1. In Windows Explorer or at a command prompt, navigate to *`SMP_HOME`*`\Servers\UnwiredServer\doe-c_clu\bin`.
2. Start up `clu.bat`.
3. Log in, or enter commands without a login.

   If you enter a command (other than **help** or **exit**) without first logging in, you must enter the DOE-C server admin listener URL, user name, and password when you are prompted for the first command that you enter. You are not prompted for this information again when you enter additional commands.

Running Commands in Batch Mode

In batch mode, the DOE-C Command Line Utility takes commands from an XML file instead of requiring you to enter them interactively through the console.

Creating an XML File to Run Commands in Batch Mode

To run commands in batch mode, you must enter them into an XML file with special tagging.

To make your batch file run smoothly:

- Use the silent option with each command. See *Using the Silent Option* on page 294.
- Specify all required parameters for each command.

1. In a text editor, create a file with the XML extension.
2. Open the file and enter these first two lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<commands>
```

3. Enter these lines for the **login** command:

```
    <command name="login" sequence="1">
        <option name="url" arg="DOECSocketListenerUrl" />
        <option name="pw" arg="DOECUserPassword" />
        <option name="u" arg="DOECUser" />
    </command>
```

4. For each command you want to execute, enter the information in an XML structure similar to this:

```
<command name="commandName" sequence="sequenceInteger">
    <option name="optionName1" arg="optionValue1" />
    <option name="optionName2" arg="optionValue2" />
    <option name="optionName3" arg="optionValue3" />
        ...
    <option name="optionNameN" arg="optionValueN" />
</command>
```

   The sequence parameter controls the order in which the commands are executed.

5. After the last <command> entry, terminate the file:

```
</commands>
```

Running the Command Line Utility in Batch Mode

The execute-commandXMLFile command runs the Command Line Utility in batch mode.

Prerequisites

Create an XML file that contains the commands that you want to execute, with proper XML tagging.

Task

1. At a command prompt, navigate to *SMP_HOME*\Servers\UnwiredServer\doe-c_clu\bin.
2. Enter:

```
execute-commandXMLFile -f xmlFileName
```

   where *xmlFileName* is either the full path to the file containing the commands to be executed, or the relative path to that file from the *SMP_HOME*\Servers\UnwiredServer\doe-c_clu\bin directory.

Using the Silent Option

Most of the commands in the DOE-C Command Line Utility support the silent option, which suppresses all user prompts, and which, in general, you want to do for batch execution.

Before using the silent option (**-sl|--silent**), verify that suppressing user prompts does not have undesirable results.

Here are some examples that illustrate potentially undesirable results:

- `getPackages -o pac.xml -sl` – if `pac.xml` already exists, it is overwritten without confirmation.
- `setPackageLogLevel -l DEBUG -i pac.xml` – if `pac.xml` contains more than one package, the log level for all packages is set to DEBUG.

Command Summary

A summary of DOE-C Command Line Utility commands. For more detailed information about a command, refer to the reference topic for the command.

Table 42. Administrative commands

Operation	Command
Start Command Line Utility console to enter commands interactively	*SMP_HOME*\Servers\UnwiredServer\doe-c_clu\bin\CLU.bat (from a command prompt)
Run Command Line Utility to take commands from an XML file	*SMP_HOME*\Servers\UnwiredServer\doe-c_clu\bin\execute-commandXMLFile.bat -f *xmlFileName* (from a command prompt)
Log in	login -u|--DOEServerUser *UnwiredServerUser* -pw|--password *UnwiredServerUserPassword* -url|--DOECSocketListenerUrl *Url* [-h|--help] [-sl|--silent]
Exit	exit [-h|--help] [-sl|--silent]
Get help	help [*commandName* | [-a|--all]] | [-h|--help]

Table 43. Package management commands

Operation	Command
Deploy a package	deploy -d|--domain *domainName* -a|--applicationID *appID* {-u|--technicalUser *SAPUserAccount* -pw|--password *SAPUserPassword*} | {-ca|-certAlias *certificateAlias*} [-sc|--securityConfiguration *securityConfigName*] [-dir|--deployFilesDirectory *deploymentDirectory*] [-h|--help] [-sl|--silent]
Get details for specific deployed packages	getPackages {-i|--in *inputXmlFile*} | {-d|--domain *domainName* [-ps|--packageNames *nameAndVersionList*]} [-o|--out *outputXmlFile*] [-h|--help] [-sl|--silent] If you omit **-ps**, **getPackages** returns a list of all deployed packages

Operation	Command
Set endpoint properties for a deployed package	`setEndpointProperties {-i|--in inputXmlFile} | {-d|--domain domainName {-a|--all | -ps|--packageNames nameAndVersionList} {-u|--technicalUser SAPUserAccount -pw|--password SAPUserPassword} | {-ca|-certAlias certificateAlias} [-ds|--doePacketDropSize byteSize] [-ew|--doeExtractWindow maxNumMsgs] [-t|--soapTimeout seconds] [-h|--help] [-sl|--silent]`
Get endpoint properties for a deployed package	`getEndpointProperties {-i|--in inputXmlFile} | {-d|--domain -a|--all | -ps|--packageNames name} [-h|--help] [-sl|--silent]`
Test endpoint properties for a deployed package	`testEndpoint {-i|--in inputXmlFile | {-d|--domain domainName -p|--packageName name}} [-h|--help] [-sl|--silent]`
Set the security configuration for deployed packages	`setPackageSecurityConfiguration {-i|--in inputXmlFile} | {-d|--domain domainName {-a|--all | -ps|--packageNames nameAndVersionList}} [-sc|--securityConfiguration securityConfigName [-h|--help] [-sl|--silent]`
Set the log level for deployed packages	`setPackageLogLevel {-i|--in inputXmlFile} | {-d|--domain domainName {-a|--all | -ps|--packageNames nameAndVersionList}} [-l|--logLevel level [-h|--help] [-sl|--silent]`
Get the log level for deployed packages	`getPackageLogLevel {-i|--in inputXmlFile} | {-d|--domain domainName {-a|--all | -ps|--packageNames nameAndVersionList}} [-h|--help] [-sl|--silent]`
Remove deployed packages	`removePackages {-i|--in inputXmlFile} | {-d|--domain domainName -ps|--packageNames name} [-h|--help] [-sl|--silent]`

Table 44. Subscription management commands

Operation	Command
Get information on subscriptions to deployed packages	`getSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -ps|--packageNames nameAndVersionList} [-o|--out outputXmlFile] [-f|--filter filterExpression] [-h|--help] [-sl|--silent]`
Get information on subscriptions to a deployed package, with sorting and pagination options	`getSubscriptions2 {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name} [-o|--out outputXmlFile] [-f|--filter filterExpression] [-pn|--pageNumber number] [-ps|--pageSize size] [-s|--sort column[:Ascending|Descending]] [-h|--help] [-sl|--silent]`
Set the log level for subscriptions to a deployed package	`setSubscriptionsLogLevel {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} -l|--logLevel level [-h|--help] [-sl|--silent]`
Get the log level for subscriptions to a deployed package	`getSubscriptionsLogLevel {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name -s|--subscriptionID ID} [-h|--help] [-sl|--silent]`
Suspend subscriptions	`suspendSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} [-h|--help] [-sl|--silent]`
Resume subscriptions	`resumeSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} [-h|--help] [-sl|--silent]`

Operation	Command
Resynchronize subscriptions to a deployed package	`resyncSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} [-h|--help] [-sl|--silent]`
End subscriptions to a deployed package	`endSubscriptions {-i|--in inputXmlFile} | {-d|--domain domainName -p|--packageName name {-a|--all | -s|--subscriptionID ID}} [-h|--help] [-sl|--silent]`

Console Management Commands

Use the administrative commands to start the Command Line Utility console, log in, get help, and exit.

login Command

Logs in to the DOE-C Command Line Utility console.

If you do not use the **login** command to log in to the Command Line Utility console, you are prompted to enter the login information for the first command (other than **help** or **exit**) that you enter.

Syntax

```
login -u|--DOEServerUser UnwiredServerUser
-pw|--password UnwiredServerUserPassword
-url|--DOECSocketListenerUrl Url
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-u|--DOEServerUser –** specifies the SAP Mobile Server admin user account.
- **-pw|--password –** specifies the SAP Control Center admin user account password.
- **-url|--DOECSocketListenerUrl –** specifies the URL for the SAP Mobile Server IIOP administration port; this is the same port specified by the `sup.admin.port` attribute in the `sup.properties` file. This port is set during installation of SAP SAP Mobile Platform.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

exit Command
Closes the Command Line Utility console.

Syntax

```
exit [-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

help Command
Displays help text for any specific DOE-C command, or for all commands.

Syntax

```
help [commandName | [-a|--all]] | [-h|--help]
```

Parameters

- **-h|--help –** gets help on the **help** command.
- **-a|--all –** gets help on all commands.
- ***commandName* –** gets help on the specified command.

Aborting Commands
There are two ways to abort commands in interactive mode.

Abort method	Description
Press Ctrl+C at any time	Aborts in-progress command and exits from Command Line Utility console. • Works at any time with any command. • To enter additional commands after using this option, you must restart the console.
Enter "abort" when prompted	Aborts in-progress command without exiting from Command Line Utility console. • Works only with certain commands, and only when prompted. • After using this option, you can continue entering commands.

Package Management Commands

Manage DOE-C packages from the Command Line Utility, rather than from SAP Control Center.

deploy Command

Deploys a DOE-C package to SAP Mobile Server.

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

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

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

Syntax

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

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-a|--applicationID –** specifies the application ID.
- **-dir|--deployFilesDirectory –** specifies the directory location that contains deployment files.
- **-sc|--securityConfiguration –** specifies the name of the security configuration to use, as defined in SAP Control Center.
- **-u|--technicalUser –** specifies the SAP technical user account to use when sending non-client-based requests. If `technicalUser` is specified, you must also specify `password`, and you must not specify `certAlias`.
- **-pw|--password –** specifies the SAP technical user account password. Required if `technicalUser` is specified.
- **-ca|--certAlias –** specifies the certificate alias. If `certAlias` is specified, you must not specify `technicalUser`.

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getPackages Command

Generates a list of deployed DOE-C packages, or returns detailed information for one or more specified packages.

Syntax

```
getPackages
{-i|--in inputXmlFile} |
{-d|--domain domainName

[-ps|--packageNames nameAndVersionList]}
[-o|--out outputXmlFile]
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. Generate the XML file using the **-o** parameter.
- **-o|--out –** saves command output to an XML file.
- **-ps|--packageNames –** specifies one or more package names for which detailed information is returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -ps myPkg1:2.0,myPkg2:1.0
  ```

  Note: If you omit this parameter, **getPackages** returns a list of all deployed packages.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

setEndpointProperties Command

Sets the DOE endpoint properties for deployed DOE-C packages.

You must set some of the DOE endpoint properties so that DOE-C can communicate with the SAP server. You can set other DOE endpoint properties to tune the performance of the communications.

Syntax

```
setEndpointProperties
{-i|--in inputXmlFile} |
{-d|--domain domainName
{-a|--all | -ps|--packageNames nameAndVersionList}
{-u|--technicalUser SAPUserAccount
```

```
-pw|--password SAPUserPassword} |
{-ca|-certAlias certificateAlias}
[-ds|--doePacketDropSize byteSize]
[-ew|--doeExtractWindow maxNumMsgs]
[-t|--soapTimeout seconds]
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-a|--all –** sets endpoint properties for all deployed packages.
- **-ps|--packageNames –** specifies one or more package names for which endpoint properties are returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -ps myPkg1:2.0,myPkg2:1.0
  ```
- **-ds|--doePacketDropSize –** the size, in bytes, of the largest JavaScript Object Notation (JSON) message that the DOE connector processes on behalf of a JSON client. The default is 1048576 bytes (1MB). The packet drop threshold size should be carefully chosen, so that it is larger than the largest message sent from the DOE to the client, but smaller than the maximum message size which may be processed by the client. Messages larger than the packet drop threshold size cause the subscription to enter the DOE packet drop state and become unusable.

  Note: Do not set lower than 4096.
- **-ew|--doeExtractWindow –** specifies the number of messages allowed in the DOE extract window. When the number of messages in the DOE extract window reaches 50% of this value, DOE-C sends a `StatusReqFromClient` message to advise the SAP DOE system of the client's messaging status and acknowledge the server's state. The default value is 50.
- **-u|--technicalUser –** specifies the SAP technical user account to use when sending non-client-based requests.
- **-pw|--password –** specifies the SAP technical user account password.
- **-ca|--certAlias –** specifies the certificate alias. If `certAlias` is specified, you must not specify `technicalUser`.
- **-t|--soapTimeout –** specifies the DOE SOAP timeout value, in seconds, to use when sending messages to the SAP DOE.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getEndpointProperties Command

Gets the DOE endpoint properties (**HTTPTimeout** value) for a deployed DOE-C package.

Syntax

```
getEndpointProperties
{-i|--in inputXmlFile} |
{-d|--domain -a|--all | -ps|--packageNames name}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads command input from an XML file.
- **-a|--all –** returns endpoint properties for all deployed packages.
- **-ps|--packageNames –** specifies one or more package names for which endpoint properties are returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -ps myPkg1:2.0,myPkg2:1.0
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

testEndpoint Command

Tests the DOE endpoint accessibility for a deployed DOE-C package.

Use the **testEndpoint** command to verify that the DOE endpoint is accessible using the parameters you set with the **setEndpointProperties** command.

Syntax

```
testEndpoint
{-i|--in inputXmlFile | {-d|--domain domainName
-p|--packageName name}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-p|--packageName –** specifies package name for which endpoint properties are set. Package name is followed by a colon and the package version number, with no white space; for example:

```
-p myPkg:2.0
```

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

setPackageSecurityConfiguration Command

Sets the security configuration for a deployed DOE-C package.

Syntax

```
setPackageSecurityConfiguration
{-i|--in inputXmlFile} |
{-d|--domain domainName
{-a|--all | -ps|--packageNames nameAndVersionList}}
[-sc|--securityConfiguration securityConfigName
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-sc|--securityConfiguration –** specifies the security configuration name, defined in SAP Control Center, to use with the specified packages.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-a|--all –** sets the security configuration for all deployed packages.
- **-ps|--packageNames –** specifies one or more package names for which the security configuration is set. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -ps myPkg1:2.0,myPkg2:1.0
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

setPackageLogLevel Command

Sets the log level, which determines the amount of information logged, for one or more deployed DOE-C packages.

Syntax

```
setPackageLogLevel
{-i|--in inputXmlFile} |
{-d|--domain domainName
{-a|--all | -ps|--packageNames nameAndVersionList}}
[-l|--logLevel level
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-l|--logLevel –** specifies the log level to be set:
  - **OFF** – no information is logged.
  - **ERROR** – only error messages are logged.
  - **WARN** – adds less serious warnings to information logged by ERROR.
  - **INFO** – adds informational messages to information logged by WARN.
  - **DEBUG** – provides the maximum amount of detail that can be logged.
- **-a|--all –** sets the log level for all deployed packages.
- **-ps|--packageNames –** specifies one or more package names for which the log level is set. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getPackageLogLevel Command

Gets the log level for one or more deployed DOE-C packages.

Syntax

```
getPackageLogLevel
{-i|--in inputXmlFile} |
{-d|--domain domainName
{-a|--all | -ps|--packageNames nameAndVersionList}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the -o parameter with the getPackages command.
- **-ps|--packageNames –** specifies one or more package names for which detailed information is returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -p myPkg:2.0
  ```

Note: If you omit **-ps**, **getPackageLogLevel** returns a list of log levels for all deployed packages.

- **-h|--help –** gets help on this command.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

removePackages Command

Removes one or more deployed DOE-C packages from the SAP Mobile Server.

Syntax

```
removePackages
{-i|--in inputXmlFile} |
{-d|--domain domainName -ps|--packageNames name}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the -o parameter with the getPackages command.
- **-ps|--packageNames –** specifies one or more package names to be removed. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -p myPkg:2.0
  ```

  Note: If you omit **-ps**, **removePackages** prompts interactively for package names to be removed.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

Subscription Management Commands

Manage DOE-C package subscriptions from the Command Line Utility, rather than from SAP Control Center.

getSubscriptions Command

Gets information on subscriptions to one or more deployed DOE-C packages.

Syntax

```
getSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-ps|--packageNames nameAndVersionList}
[-o|--out outputXmlFile]
[-f|--filter filterExpression]
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-o|--out –** saves command output to an XML file.
- **-f|--filter –** specifies the filter to use on the subscriptions. Each column name is followed by a colon and the filter string. Use a comma to separate the information for multiple column names, with no white space; for example:

  ```
  -f columnName:filterString,columcName2:filterString2
  ```

  Valid filter column names are: subscriptionID, packageName, clientID, physicalID, logicalID, userName, language, clientMsgID, clientMsgTimeStamp, serverMsgID, serverMsgTimeStamp, logLevel.

  You can use "?" and "*" wildcard characters in your filter strings; for example:

  ```
  -f clientMsgTimeStamp:*Jan*21?41*2009,userName:john*
  ```

- **-ps|--packageNames –** specifies one or more package names for which subscription information is returned. Each package name is followed by a colon and the package version number. Use a comma to separate the information for multiple packages, with no white space; for example:

  ```
  -p myPkg:2.0
  ```

  Note: If **-ps** is omitted, **getSubscriptions** prompts you for a package name.

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getSubscriptions2 Command

Gets information on subscriptions to a deployed DOE-C packages, with output paginated and sorted.

Syntax

```
getSubscriptions2
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name}
[-o|--out outputXmlFile]
[-f|--filter filterExpression]
[-pn|--pageNumber number] [-ps|--pageSize size]
[-s|--sort column[:Ascending|Descending]]
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.

- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with **getPackages**.
- **-o|--out –** saves command output to an XML file.
- **-f|--filter –** specifies the filter to use on the subscriptions. The filter expression must have one column name, followed by a colon and the filter string; for example:

  ```
  -f columnName:filterString
  ```

  Valid filter column names are: subscriptionID, packageName, clientID, physicalID, logicalID, userName, language, clientMsgID, clientMsgTimeStamp, serverMsgID, serverMsgTimeStamp, logLevel.

  You can use "?" and "*" wildcard characters in your filter strings; for example:

  ```
  -f clientMsgTimeStamp:*Jan*21?41*2009
  ```
- **-p|--packageName –** specifies package name for which subscription information is returned. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-ps|--pageSize –** specifies the page size, which is the number of subscriptions per page returned. Page size must be 1 or higher. If you do not specify a page size:
  - If the number of subscriptions returned is greater than 10, you are prompted to enter a page size.
  - If the number of subscriptions returned is 10 or fewer, all subscriptions are listed on one page.
- **-pn|--pageNumber –** specifies the page number, which is the number of the page returned, determined by the page size. Page number must be 1 or higher. If you do not specify a page number:
  - If the number of subscriptions returned is greater than the page size, you are prompted to enter a page number.
  - If the number of subscriptions returned is not greater than the page size, all subscriptions are listed on one page.

  Page size and page number together determine the subscriptions actually returned by for the specified package name; for example, you might specify a page size of 3 with a page number of 2:

  ```
  getSubscriptions2 -p myPkg:2.0 -ps 3 -pn 2
  ```

  This example returns the second page of subscriptions for version 2.0 of the package named myPkg. That page would contain subscriptions 4-6 to the package. With a page size of 3, the fist page would contain subscriptions 1-3, the third page would contain subscriptions 7-9, and so on. If sorting or filtering are specified, these operations produce the list of subscriptions to which page size and page number are applied.
- **-s|--sort –** specifies the columns on which output is to be sorted. If you specify only a column name, the default sort order is ascending; for example:

```
-s UserName
```

Add a colon, followed by `Descending` after the column name to sort in descending order; for example:

```
-s ServerMsgTimeStamp:Descending
```

Valid sort column names are: ClientID, PhysicalID, SubscriptionID, LogicalID, PushQueue, UserName, Language, LogLevel, ServerMsgID, ServerMsgTimeStamp, ClientMsgID, ClientMsgTimeStamp, ApplicationName, and MMSPID.

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

setSubscriptionsLogLevel Command

Sets the log level, which determines the amount of information logged, for subscriptions to a deployed DOE-C package.

Syntax

```
setSubscriptionsLogLevel
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name {-a|--all |
-s|--subscriptionID ID}}
-l|--logLevel level
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the -o parameter with the getPackages command.
- **-l|--logLevel –** specifies the log level to be set:
  - **OFF** – no information is logged.
  - **ERROR** – only error messages are logged.
  - **WARN** – adds less serious warnings to information logged by ERROR.
  - **INFO** – adds informational messages to information logged by WARN.
  - **DEBUG** – provides the maximum amount of detail that can be logged.
- **-a|--all –** specifies the log level for all deployed packages.
- **-p|--packageName –** specifies a package name for which the log level is set. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs for which you want to set the log level. Use a comma to separate multiple subscription IDs, with no white space; for example:

```
-s mySubs1,mySubs2
```

- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

getSubscriptionsLogLevel Command

Gets the log level for subscriptions to a deployed DOE-C package.

Syntax

```
getSubscriptionsLogLevel
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
-s|--subscriptionID ID}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads command input from an XML file.
- **-p|--packageName –** specifies a package name for which detailed information is returned. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs for which you want to get the log level. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```

  Note: If **-s** is omitted, **getSubscriptionsLogLevel** returns a list of all subscriptions for the specified package.
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

suspendSubscriptions Command

Use the **suspendSubscriptions** command to suspend subscriptions to one or all deployed DOE-C packages.

Syntax

```
suspendSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
{-a|--all | -s|--subscriptionID ID}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads subscription ID and package name from input XML file. You can generate the XML file by using the **-o** parameter with the **getSubscriptions** command.
- **-a|--all –** suspends subscriptions for all deployed packages.
- **-p|--packageName –** specifies a package name for which subscriptions are suspended. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs which you want to suspend. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```
- **-sl|--silent –** disables all user interactive questions; this option is mainly used when writing a batch file.

resumeSubscriptions Command

Use the **resumeSubscriptions** command to resume subscriptions to one or all deployed DOE-C packages for which subscriptions have been suspended.

Syntax

```
resumeSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
{-a|--all | -s|--subscriptionID ID}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads subscription ID and package name from input XML file. You can generate the XML file by using the **-o** parameter with the **getSubscriptions** command.
- **-a|--all –** resumes subscriptions for all deployed packages.
- **-p|--packageName –** specifies a package name for which subscriptions are resumeed. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs which you want to resume. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```

- **-sl|--silent –** disables all user interactive questions; this option is mainly used when writing a batch file.

resyncSubscriptions Command

Unblocks DOE queues.

If the SAP DOE Connector does not respond to the SAP DOE quickly enough, the DOE may mark that subscription's queues as "blocked" and stop sending messages to the DOE-C. At start-up, the DOE-C sends a RestartQ message to the DOE that should unblock these queues. If this happens at times other than at start-up, you can use **resyncSubscriptions** to resume communication from the DOE to the DOE-C.

Syntax

```
resyncSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
{-a|--all | -s|--subscriptionID ID}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with the **getPackages** command.
- **-a|--all –** reactivates subscriptions for all deployed packages.
- **-p|--packageName –** specifies package name for which subscriptions are reactivated. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs to recover. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

endSubscriptions Command

Ends subscriptions to a deployed DOE-C package.

Syntax

```
endSubscriptions
{-i|--in inputXmlFile} |
{-d|--domain domainName
-p|--packageName name
```

```
{-a|--all | -s|--subscriptionID ID}}
[-h|--help] [-sl|--silent]
```

Parameters

- **-h|--help –** gets help on this command.
- **-d|--domain –** specifies the domain.
- **-i|--in –** reads package name from input XML file. You can generate the XML file by using the **-o** parameter with the **getPackages** command.
- **-a|--all –** ends subscriptions for all deployed packages.
- **-p|--packageName –** specifies package name for which subscriptions are ended. Package name is followed by a colon and the package version number, with no white space; for example:

  ```
  -p myPkg:2.0
  ```
- **-s|--subscriptionID –** specifies one or more subscription IDs to recover. Use a comma to separate multiple subscription IDs, with no white space; for example:

  ```
  -s mySubs1,mySubs2
  ```
- **-sl|--silent –** disables all user interactive questions; this option is generally used with batch files.

Agentry Utilities

Use the Agentry quick password utility to manage password encoding for Agentry applications. Launch this utility from the command line; it is not available from SAP Control Center.

Quick Password Utility Overview

The Quick Password Utility is used to encode the various passwords found in the `Agentry.ini` configuration file of the Agentry Server. Each configuration section containing a password value can have that password encoded by this utility.

Running the encoder on the command line results in a series of prompts, one for each of the passwords to encode. You may choose to encode the given password or not. If you elect to do so, the password value for the section is encoded. Additionally, each section has a configuration option indicating whether the password value is encoded. This option is set to `false` by default. The password encoder utility can set this configuration option to `true`.

The following table contains a list of the configuration sections, the password option, and the encoded flag that the quick password utility modifies.

Configuration Section	Password	Encoded Flag
[Angel Front End]*	authenticationCertificateStore-Password	authenticationCertificateStorePasswor-dEncoded

Configuration Section	Password	Encoded Flag
[Web Server Front End]*	authenticationCertificateStore-Password	authenticationCertificateStorePasswor-dEncoded
[SQL-n]	dbConnectionPassword	dbConnectionPasswordEncoded
[HTTPXML-n]	basicAuthenticationPassword	basicAuthenticationPasswordEncoded

*- *These passwords are only encoded if the configuration option* `authenticationCertificateStore` *is set to a value other than the default of* `AgentryServer.pfx`. *The password for this default certificate is encoded by default when the Server is installed. Changing the certificate will require the utility to be used to encrypt the password for the new certificate*.

Using the Quick Password Encoding Utility

Prerequisites

Prior to running the quick password utility, address the following items:

- Determine which passwords to encode that are in the `Agentry.ini` file.
- Obtain the proper passwords for all you are encoding, as you are required to enter them when running the encoder utility.
- Determine the best time to restart the Agentry Server after this change is made.
- If using the `AgentryServer.pfx` file for either the ANGEL or Web Server Front End communications, the password for this certificate file cannot be encoded. The utility will not prompt you for the related password values.

Task

The quick password utility is run from the Windows command prompt and from within the folder where the Agentry Server is installed. The utility is run through the command `quickPW`, which takes no arguments. The user running the command must have read-write access for the Server's folder and all its contents, particularly the `Agentry.ini` configuration file.

For each section in which a password is contained, this utility will display a prompt to encode the password. A 'y' or 'n' response is expected for yes or no, respectively. If the 'y' response is entered, an additional prompt is displayed for the plain text password to be entered. the password is then encoded and written to the `Agentry.ini` file. Additionally, each password setting also has a corresponding encoded flag setting that indicates whether or not the password is encoded. This is required in order for the Agentry Server to properly process the password value when it is read in. The utility will set this flag to true for each password it encodes.

If an 'n' response is received for any section, the password is not changed and the utility moves on to the next section until all passwords within the `Agentry.ini` file have been processed.

1. Save a copy of the current `Agentry.ini` file to a backup location prior to running this command.
2. Open a Windows (DOS) command prompt and navigate to the location of the Agentry Server instance for the mobile application, i.e., `C:\SAP\MobilePlatform \Servers\UnwiredServer\<AppName>`.
3. Enter the command: `quickPW`.

   The following prompt displays. Note that if one of the following sections is not a part of the `Agentry.ini` file, the prompt for it will not be displayed, where *`Section Name`* corresponds to the section with a password to be encoded:

   `Configure '`*`Section Name`*`'? [Y/n]`
4. If you want to encode the password for this section, enter the character `Y`. Otherwise, enter `n` to skip this section.

   If `Y` is entered, the following prompt displays.

   ```
   Please enter new password:
   ```
5. Type the new password after this prompt and press **Enter**.

   The password for the section is encoded and the corresponding password encoded flag option is set to `true`. You are then prompted to encode the password for the next section.
6. Repeat this procedure until you have processed all sections within the `Agentry.ini` file.

   After all sections are processed, the utility exits and you are returned to the Windows command prompt.
7. Restart the Agentry Server if it is running in order for the modifications to take effect.

The password values you chose to encode are encoded within the `Agentry.ini` file. These values are no longer human-readable. The encoded password flag option that pertains to each encoded password is set to `true`.

Configuration Files

Configuration files hold the initial settings for various SAP Mobile Platform components or subcomponents.

All SAP Mobile Platform components read their configuration files at start-up. Some components check configuration files for changes periodically. Administrators can instruct SAP Mobile Server to reread configuration files, and apply changes to the current process. However, some configuration changes require you to restart the SAP Mobile Server.

SAP Mobile Server Configuration Files

Use SAP Mobile Server configuration files to manually modify server security, logging, and subcomponent configurations.

SAP recommends that you edit these `.properties` and `.xml` files only if you cannot use SAP Control Center to configure SAP Mobile Server properties.

See also

- *SAP Control Center Configuration Files* on page 321
- *Relay Server Configuration Files* on page 324
- *Data Tier Configuration Files* on page 337
- *Agentry Configuration Files* on page 338

Admin Security (default.xml) Configuration File Reference

Defines authentication and attribution properties for the 'admin' security configuration. The file is located in *SMP_HOME*`\Servers\UnwiredServer\Repository\CSI\conf`.

Note: SAP recommends that you use SAP Control Center to configure security settings, so that configuration changes are validated before being saved.

Define the login modules used for authentication requests. List the modules in the order that they are invoked with this syntax:

```
<config:authenticationProvider name="<authProviderName>"
controlFlag="<myValue>" />
```

See the *controlFlag Attribute Values* reference topic for possible configuration values. The default is `required`.

Configure global options if the same configuration is shared by the authentication and attribution providers by using:

```
<config:options name="<propertyName>" value="<myValue>" />
```

For an LDAP security provider, properties can include:

Property	Default	Description
ServerType	None	The LDAP server type. For example, `msad2k`, `sunone5`, `nsds4`, or `openldap`.

Property	Default	Description
ProviderURL	`ldap://<host-name>:389`	The URL to connect to the LDAP server. For example, `ldap://localhost:389`. For SSL, use `ldap://localhost:636`.
SecurityProtocol	None	The protocol used to connect to the LDAP server. SAP recommends that you set this to `SSL`. An SSL connection is required for Active-Directory when you set the password attribute.
InitialContextFactory	None	The factory class used to obtain initial directory context.
Referral	ignore	The behavior when a referral is encountered. The valid values are those dictated by LdapContext, for example, `ignore`, `follow`, or `throw`.
DefaultSearchBase	None	The default search base to use when performing general operations.
AuthenticationSearchBase	None	The search base to use when performing authentication operations. If you do not set this value, the default search base is used.
SelfRegistrationSearchBase	None	The search base to use when creating a new user as part of self-registration. If you do not set this value, the authentication search base is used for self-update operations and the default search base is used for all other operations.
AuthenticationScope	ONELEVEL	Options include `ONELEVEL` or `SUBTREE`.

Property	Default	Description
AuthenticationFilter	For most LDAP servers: `(&(uid={uid})(objectclass=person))` or For Active Directory e-mail lookups: `(&(userPrincipalName={uid})(objectclass=user)) [ActiveDirectory]` For Active Directory Windows user name lookups: `(&(sAMAccountName={uid})(objectclass=user))`	The user name and password authentication search filter. This must be a legal LDAP search filter, as defined in RFC 2254. The filter may contain the special string `"{uid}"` which is replaced with the user name of the user attempting to authenticate.
CertificateAuthentication-Filter	For Active Directory server: `(&({certattr}={0})(objectclass=user))"` For most LDAP server types: `"(&({certattr}={0})(objectclass=person))"`	The certificate authentication search filter. The filter may contain the special string `"{certattr}"` which is replaced with the certificate attribute or the mapped LDAP attribute (if mapping between certificate attributes and LDAP attributes is defined). The value `"{0}"` is set to the encoded certificate or an attribute value from the certificate.
AuthenticationMethod	simple	The authentication method to use for all binding. Supported values are `DIGEST-MD5` or `simple`.
Attributes	None	Defines an attribute mapping from a CSI attribute to an LDAP attribute, including: • `CSI.Email` • `CSI.Username` • `CSI.Password`

Property	Default	Description
BindDN	None	The DN to bind against when building the initial LDAP connection. The user being authenticated with the login module must have read permission on all user records.
BindPassword	None	The password to bind with for the initial LDAP connection. For example, <config:options name="BindPassword" encrypted="true" value="1-AAAAEgQ-QyyYzC2+njB4K4QGPcMB1pM6XErTqZ1InyYrW/s56J69VfW5iBdFZeh-DrY66+6g9u1+a5VAqBiv/v5q08B3f59YMB1EQx9k93VgVTSC0w8q0="/>.
RoleSearchBase	None	The search base used to retrieve lists of roles. If this you do not set this value, the default search base is used.
RoleFilter	For SunONE/iPlanet: (&(objectclass=ldapsubentry) (objectclass=nsroledefinition)) For Netscape Directory Server: (objectclass=groupofnames) (objectclass=groupofuniquenames)) For ActiveDirectory: (objectclass=groupofnames) (objectclass=group))	When combined with the role search base and role scope, returns a complete list of roles within the LDAP server.

Property	Default	Description
RoleScope	ONELEVEL	The role search scope. Options include `ONELEVEL` or `SUBTREE`.
RoleNameAttribute	cn	The attribute for retrieved roles that is the common name of the role.
UserFreeformRoleMembershipAttributes	None	The "freeform" role membership attribute list. Users who have attributes in this comma-delimited list are automatically granted access to roles that have names that are the same as the attribute value.
UserRoleMembershipAttributes	The default value of this property depends on the chosen server type: • For SunONE 5.x, it is `"nsRoleDN"` • For ActiveDirectory, it is `"memberOf"`.	Defines the attributes that contain the DNs of all of the roles the user is a member of. These comma-delimited values are then cross-referenced with the roles retrieved from the role search base and search filter to create a list of user roles.
RoleMemberAttributes	There is a default value only for Netscape 4.x server: `"member,uniquemember"`	A comma-delimited list of potential attributes for roles. This list defines the DNs of people who are granted the specified roles. These values are cross-referenced with the active user to determine the user's roles.

Configure additional security providers using:

```
<config:provider name="<secProviderName>" type="<secType>" />
```

Security types include: `authorizer`, `attributer`, or `roleMapper`.

controlFlag Attribute Values

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

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

Control Flag Value	Description
Required	The LoginModule is required. Authentication proceeds down the LoginModule list.

Control Flag Value	Description
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 (default)	The LoginModule is not required. Regardless of success or failure, authentication proceeds down the LoginModule list.

Example

Providers are listed in this order and with these controlFlag:

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

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

SAP Control Center Configuration Files

Use SAP Control Center configuration files to configure SAP Control Center services, plug-ins, logging, and security.

SAP recommends that you edit these `.properties` and `.xml` files only if you cannot use SAP Control Center to configure the properties.

See also

- *SAP Mobile Server Configuration Files* on page 316

- *Relay Server Configuration Files* on page 324
- *Data Tier Configuration Files* on page 337
- *Agentry Configuration Files* on page 338

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

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

Key `service-config.xml` files include:

File	Description	Defaults
SCC_HOME\services\RMI\service-config.xml	Sets the RMI agent port.	RMI port: 9999
SCC_HOME\services\Messaging\service-config.xml	Sets the JMS messaging service port.	JMS messaging port: 2100
SCC_HOME\services\SccSADataserver\service-config.xml	Sets the SAP Control Center repository database port, and other properties.	Repository database port: 3683
SCC_HOME\services\EmbeddedWebContainer\service-config.xml	Sets the Web container ports.	HTTP port: 8282 HTTPS port: 8283

Agent Plug-in Properties (agent-plugin.xml) Configuration File

Defines server properties for each SAP Mobile Server to administer from SAP Control Center. The file is located in *SCC_HOME*\plugins\com.sybase.supadminplugin.

Properties include:

Property	Default	Description
auto.discovery.enable.on.startup	Personal Developer Edition: false Other editions: true	Enables or disables the automatic discovery of SAP Mobile Servers when SAP Control Center starts.

Property	Default	Description
auto.discovery.log.repeat	3	Repeats the logging of errors that are generated when a discovered SAP Mobile Server is pinged. After the specified count is reached, SAP Control Center *X.X* stops printing error message to the log, but continues to ping the discovered server.
auto.discovery.schedule.enable.on.startup	true	Enables or disables scheduled SAP Mobile Server discoveries. Scheduled discovery allows the agent to periodically check for newly installed SAP Mobile Servers.
auto.discovery.schedule.interval	300000	Sets the scheduled discovery interval (in milliseconds).
sup.server.path	*SCC_HOME*\Servers\UnwiredServer	Sets the installation for SAP Mobile Server. The path must be fully-qualified.
auto.register.localSUP.enable	true	If true, automatically registers the server as a managed resouce in SAP Control Center. After launching and authenticating, registered resources automatically appear in the SAP Mobile Platform. If false, you need to manually register the server as managed resource in SAP Control Center.
server.edition	none	The local SAP Mobile Server edition: • ED – enterprise developer edition • PD – personal developer edition

SAP Control Center Logging (log4j.properties) Configuration File

Enables and configures SAP Control Center logging. The log4j configuration file is located in *<UnwiredPlatform_InstallDir>*\SCC-*XX*\conf.

log4j properties include:

Property	Default	Description
log4j.appender.agent.File	${com.sybase.ua.home}/log/agent.log	The name and location of the SAP Control Center log file.
log4j.appender.agent.MaxFileSize	25MB	The maximum size that a file can reach before a new one is created.
log4j.appender.agent.MaxBackupIndex	20	The number of log files that are backed up before the oldest file is deleted.

Relay Server Configuration Files

Use Relay Server configuration files to set up Relay Servers. Use RSOE configuration files to set up Outbound Enablers.

See also

- *SAP Mobile Server Configuration Files* on page 316
- *SAP Control Center Configuration Files* on page 321
- *Data Tier Configuration Files* on page 337
- *Agentry Configuration Files* on page 338

Relay Server Configuration (rs.config)

The `rs.config` file defines the configuration of individual Relay Servers, and Relay Server clusters (farms).

The rs.config file is divided into four sections:

- **[relay_server] –** defines configuration for a single Relay Server
- **[backend_farm] –** defines a homogeneous group (e.g., a cluster) of back-end servers, which are targeted by client requests
- **[backend_server] –** defines the connection to each back-end server, supported by one or more RSOEs
- **[options] –** defines general configuration properties that apply to all Relay Servers in a cluster (farm)

[relay_server] Section

- **enable –** Specifies whether the Relay Server is included in a Relay Server cluster.

  Possible values are:

  - yes
  - no

  Default is `yes`.
- **host –** Host name or IP address to be used by the Outbound Enabler to make a direct connection to the Relay Server.
- **http_port –** HTTP port to be used by the Outbound Enabler to make a direct connection to the Relay Server.

  Possible values are:

  - 0 or off — Disable HTTP access from Outbound Enabler
  - 1 to 65535 — Enable HTTP access on the specified port

  Default is `80`.
- **https_port –** HTTPS port to be used by the Outbound Enabler to make a direct connection to the Relay Server.

  Possible values are:

  - 0 or off — Disable HTTP access from Outbound Enabler
  - 1 to 65535 — Enable HTTP access on the specified port

  Default is `443`.
- **description –** User-definable description of the Relay Server, up to 2048 characters.

[backend_farm] Section

- **active_cookie –** Specifies whether a cookie is set to maintain client-server session affinity.

  Possible values are:

  - yes — Relay Server injects a standard HTTP set-cookie command, with a proprietary cookie name in the response.
  - no — Use this option when the back-end farm serves a sessionless browser application.
- **active_header –** Specifies whether a header is set to maintain client-server session affinity.

  Possible values are:

  - yes — Relay Server injects a proprietary header in the response, in case intermediaries tamper with the active cookie.

    - no — Use this option to reduce traffic volume, if the back-end farm serves only browser applications, or if the active cookie works for all clients of the back-end farm.
- **backend_security –** Specifies the level of security required of an Outbound Enabler in the back-end farm.

  Possible values are:

  - on — All connections from the back-end farm must use HTTPS.
  - off — All connections from the back-end farm must use HTTP.

  If no value is specified, the Outbound Enabler can use either HTTP or HTTPS.
- **client_security –** Specifies the level of security the back-end farm requires of its clients.

  Possible values are:

  - on — All clients must use HTTPS.
  - off — All clients must use HTTP.

  If no value is specified, the clients can use either HTTP or HTTPS.
- **description –** User-definable description of the back-end farm, up to 2048 characters.
- **enable –** Specifies whether to allow connections from the back-end farm.

  Possible values are:

  - yes
  - no

  Default is `yes`.
- **id –** User-definable string that identifies the back-end farm, up to 2048 characters.
- **verbosity –** Relay Server logging verbosity.

  Possible values are:

  - 0 — Log errors only.
  - 1 — Session-level logging.
  - 2 — Request-level logging.
  - 3 - 5 — Detailed logging, used primarily for technical support.

  Default is `0`.

[backend_server] Section

- **description –** User-definable description of the back-end server, up to 2048 characters.
- **enable –** Specifies whether to allow connections from the back-end server.

  Possible values are:

  - yes
  - no

Default is `yes`.

- **farm –** User-definable string that identifies a back-end farm.

  This property associates the back-end server with a back-end farm. This value must match the id value in a [backend_farm] section.
- **id –** User-definable string that identifies the back-end server, up to 2048 characters.
- **MAC –** MAC address of the network adapter used by Outbound Enabler to make a connection with the Relay Server.

  If this property is not specified, Relay Server does not check the MAC address on Outbound Enabler connections.
- **token –** A security token used by Relay Server to authenticate the back-end server connection, up to 2048 characters.
- **verbosity –** Relay Server logging verbosity.

  Possible values are:

  - 0 — Log errors only.
  - 1 — Session-level logging.
  - 2 — Request-level logging.
  - 3 - 5 — Detailed logging, used primarily for technical support.

  Default is `0`.

[options] Section

- **start –** Method used to start the State Manager.

  Possible values are:

  - auto — State Manager is started automatically by Relay Server Web extensions.
  - no — State Manager is started as a service.
  - full path — Full path to the State Manager executable (`rshost`).
  -

  Default is `auto`.
- **shared_mem –** Maximum amount of shared memory used for state tracking.

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

  Possible values are:

  - 0 — Log errors only.
  - 1 — Session-level logging.

- 2 — Request-level logging.
- 3 - 5 — Detailed logging, used primarily for technical support.

Default is `0`.

Example: N+2 Architecture Configuration

There are two Relay Servers in a Relay Server farm: RS1.sybase.com and RS2.sybase.com. There are also two back-end server types: Afaria and SAP Mobile Server. The Afaria system has two servers: Afaria1.sap.com and Afaria2.sap.com. There is one SAP Mobile Server: SMP.sap.com.

Relay Servers use Microsoft IIS, and the Web server is configured with the following paths:

- `\ias_relay_server\client` – the install location of `rs_client.dll`.
- `\ias_relay_server\server` – the install location of `rs_server.dll`, `rshost.exe`, and `rs.config`.

`#----------------------------`
`# Relay server with auto start option`
`#--------------------------------`
`------`
`[options]`
`start = no`
`verbosity = 1`
	Start State Manager as a Windows service.

`#--------------------`
`# Relay server peers`
`#--------------------`
`[relay_server]`
`enable = yes`
`host = RS1.sap.com`
`http_port = 80`
`https_port = 443`
`description = Machine #1 in RS farm`
 	
`[relay_server]`
`enable = yes`
`host = RS2.sap.com`
`http_port = 80`
`https_port = 443`
`description = Machine #2 in RS farm`
	Because there are two Relay Servers, both instances need to be defined in the peer list. This list is used by the RSOE to establish a connection with each Relay Server node in the farm.

`#---------------`
`# Backend farms`
`#---------------`
`[backend_farm]`
`enable = yes`
`id = AfariaFarm`
`client_security = on`
`backend_security= on`
`description = This is the definition for the Afaria farm.`
 	
`[backend_farm]`
`enable = yes`
`id = SMPFarm`
`client_security = on`
`backend_security= on`
`description = This is the definition for the SMP farm.`
	There are two types of back-end farms in this example. SAP Mobile Server and Afaria. Each farm requires an entry in the [backend_farms] section, and each must have a unique Farm ID.

`#-----------------`
`# Backend servers`
`#-----------------`
`[backend_server]`
`enable = yes`
`farm = AfariaFarm`
`id = for Afaria1.sap.com`
`mac = 01-23-45-67-89-ab`
`token = 7b2493b0-d0d4-464f-b0de-24643e1e0feb`
 	
`[backend_server]`
`enable = yes`
`farm = AfariaFarm`
`id = for Afaria1.sap.com`
`mac = 01-23-45-67-89-ac`
`token = de1aac83-a653-4e0f-8a6c-0a161a6ee407`
 	
`[backend_server]`
`enable = yes`
`farm = SMPFarm`
`id = for SMP.sap.com`
`mac = 01-23-45-67-89-ad`
`token = 621ece03-9246-4da7-99e3-c07c7599031c`
	There are three back-end server nodes in this scenario. Two are part of the AfariaFarm and the third is part of the SMPFarm. Afaria back-end servers use the Transmitter ID as the Server ID. SAP Mobile Platform typically uses the machine name as the Server ID. The Server ID must be unique for each back-end server entry.

Note: When Relay Servers have been deployed and configured, clients connect to severs in the back-end farms differently:

- For the AfariaFarm example, clients would use:
  `url_suffix=/ias_relay_server/client/rs_client.dll/AfariaFarm`
- For the SUPFarm example, clients would use:

```
url_suffix=/ias_relay_server/client/rs_client.dll/SMPFarm
```

Outbound Enabler Configuration (rsoeconfig.xml)

The `rsoeconfig.xml` file defines Outbound Enabler (RSOE) and Relay Server configurations.

This information describes the XML schema of the RSOE configuration file.

Element: relayServers

XML root element.

Parents	n/a
Model	relayServer*, proxyServer*

Attribute	Value	Required
xmlns	Type: string http://www.sybase.com/sup/SUPRelayServerConfig	Yes

Element: relayServer

Identifies a Relay Server instance.

Parents	//relayServers
Model	description{0,1}, httpCredential*, unwiredServerFarm*

Attribute	Value	Required
ID	Type: integer Primary key of the Relay Server entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
host	Type: string Host name of the Relay Server, or host name of the load balancer (if any).	Yes
port	Type: non-negative integer, 0-65535 Relay Server HTTP port.	Yes
securePort	Type: non-negative integer, 0-65535 Relay Server HTTPS port.	Yes

Attribute	Value	Required
urlSuffix	Type: string URL suffix used by an RSOE to connect to the Relay Server.	Yes

Element: proxyServer
Identifies an Internet proxy server instance.

Parents	//relayServers
Model	proxyUser*

Attribute	Value	Required
guid	Type: ID Identifies the IP address and port of the proxy server, in this form: `IPaddress_port` For example: `10.56.225.169_808`	No
ID	Type: integer Primary key of the proxy server entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
host	Type: string Host name of the proxy server.	Yes
port	Type: non-negative integer, 0-65535 Proxy server HTTP port.	Yes

Element: description
User-definable description of a Relay Server, or an SAP Mobile Server cluster.

Parents	//relayServers/relayServer //relayServers/relayServer/unwiredServerFarm
Model	n/a (text node only)

Attribute	Value	Required
n/a		

Element: httpCredential
Specifies credentials an RSOE must use to authenticate its connection to the Relay Server.

Parents	//relayServers/relayServer
Model	n/a (empty)

Attribute	Value	Required
ID	Type: integer Primary key of the credential entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
userName	Type: string User name RSOE must use to access the Relay Server.	Yes
password	Type: string Password RSOE must use to access the Relay Server.	Yes

Element: unwiredServerFarm
Identifies an SAP Mobile Server cluster.

Parents	//relayServers/relayServer
Model	description{0,1}, serverNode*

Attribute	Value	Required
ID	Type: integer Primary key of the server farm entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
name	Type: string Name of the SAP Mobile Server cluster, as known to the Relay Server.	Yes
type	Type: string, enumerated Type of SAP Mobile Server connection. Allowed values are: • messaging • replication	Yes

Element: proxyUser
Specifies credentials an RSOE must use to authenticate its connection to an Internet proxy server.

Parents	//relayServers/proxyServer
Model	n/a (empty)

Attribute	Value	Required
guid	Type: ID Identifies the IP address and port of the proxy server, with an appended user identifier, in this form: `IPaddress_port_UserId` For example: `10.56.225.169_808_User-001`	No
ID	Type: integer Primary key of the credential entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
userName	Type: string User name RSOE must use to access the proxy server.	Yes
password	Type: string Password RSOE must use to access the proxy server.	Yes

Element: serverNode
Identifies an SAP Mobile Server instance in the SAP Mobile Server cluster.

Parents	//relayServers/relayServer/unwiredServerFarm
Model	rsoe{0,1}

Attribute	Value	Required
ID	Type: integer Primary key of the server node entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
name	Type: string Name of the SAP Mobile Server instance, as known to the Relay Server.	Yes

Attribute	Value	Required
token	Type: string Token string RSOE must use to authenticate its connection to the Relay Server.	Yes

Element: rsoe

Identifies an Outbound Enabler (RSOE) associated with an SAP Mobile Server instance.

Parents	//relayServers/relayServer/unwiredServerFarm/serverNode
Model	supServerPort, clusterMember, tlsOptions{0,1}

Attribute	Value	Required
ID	Type: integer Primary key of the RSOE entry in the cluster database. If value is 0 or unspecified, a new entry is created.	No
httpUser	Type: string If specified, this value should match the value of a //relay-Server/httpCredential/@userName instance.	No
proxyServerRef	Type: IDREF If specified, this value must match the value of a //proxyServ-er/@guid instance.	No
proxyUserRef	Type: IDREF If specified, this value must match the value of a //proxyServ-er/proxyUser/@guid instance.	No
startOptions	Type: string Defines RSOE command-line startup options. If specified, this value must match the pattern: `\s*(((-v\s+[0-5])|(-d\s+\d+)|(-os\s+(1024\d|102[5-9]\d|10[3-9]\d{2}|1[1-9]\d{3}|[2-9]\d{4}|[1-9]\d{5,}))|(-ot))\s+)*((-v\s+[0-5])|(-d\s+\d+)|(-os\s+(1024\d|102[5-9]\d|10[3-9]\d{2}|1[1-9]\d{3}|[2-9]\d{4}|[1-9]\d{5,}))|(-ot))?\s*`	No

Attribute	Value	Required
useHttps	Type: boolean Specifies whether RSOE uses HTTP or HTTPS in connection to Relay Server.	Yes

Element: supServerPort
Identifies the SAP Mobile Server port managed by an RSOE.

Parents	//relayServers/relayServer/unwiredServerFarm/serverNode/rsoe
Model	n/a (empty)

Attribute	Value	Required
port	Type: non-negative integer, 0-65535	Yes

Element: clusterMember
Identifies the name of the SAP Mobile Server instance associated with the RSOE.

Parents	//relayServers/relayServer/unwiredServerFarm/serverNode/rsoe
Model	n/a (empty)

Attribute	Value	Required
name	Type: string Name of the SAP Mobile Server instance, as known to the SAP Mobile Server cluster.	Yes

Element: tlsOptions
Specifies TLS configuration for RSOE connection to Relay Server.

Parents	//relayServers/relayServer/unwiredServerFarm/serverNode/rsoe
Model	n/a (empty)

Attribute	Value	Required
certificateCompany	Type: string Organization name field of certificate.	No

Attribute	Value	Required
certificateFile	Type: anyURI Location of certificate file.	Yes
certificateName	Type: string Common name field of certificate.	No
certificateUnit	Type: string Organization unit field of certificate.	No
tlsType	Type: string, enumerated Allowed values are: • RSA	No

Content of rsoeConfig.template.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<relayServers xmlns="http://www.sybase.com/sup/
SUPRelayServerConfig">
    <!--
       If the "ID" attribute for a XML element is 0 means you want to
create
       a new element. If the "ID" attribute of a element is a positive
        integer means you want update a existing element.
    -->
    <relayServer securePort="443" port="80"
host="relayserver.sybase.com"
        urlSuffix="/ias_relay_server/server/rs_server.dll" ID="0">
       <description>Relay Server/RSOE Configuration example. A relay
            server could have many Backend Farms.
        </description>
        <unwiredServerFarm type="replication" name="farm1.myRBS"
ID="0">
           <description>Replication Backend Farm example. A Backend
Farm could have
                many Backend Servers.</description>
            <serverNode token="36413d78ef187a7b38548e00e586"
name="node1"
                ID="0">
              <!-- A Backend Server can have 0 or 1 RSOE mapping to
it. -->
             <rsoe useHttps="false" ID="0" startOptions="-ot -v 0 -d
10">
                    <supServerPort port="2480" />
                    <clusterMember name="ExampleServer1" />
                </rsoe>
            </serverNode>
            <serverNode token="123" name="node2" ID="0" />
           <serverNode token="my_secret_token" name="node3" ID="0">
```

```
               <rsoe startOptions="-v 0 -ot" useHttps="true" ID="0">
                    <supServerPort port="2480" />
                    <clusterMember name="ExampleServer3" />
                    <tlsOptions certificateFile="E:\tmp
\mms-1.6.0.1118\RsoeCerts\myserver.pem" />
                </rsoe>
            </serverNode>
        </unwiredServerFarm>
        <unwiredServerFarm type="messaging" name="farm2.myMBSFarm"
            ID="0">
            <description>test</description>
        </unwiredServerFarm>
    </relayServer>
    <relayServer securePort="443" port="80"
        host="relayserver.example.com" ID="0" urlSuffix="/srv/
iarelayserver">
        <description>test</description>
    </relayServer>
</relayServers>
```

Data Tier Configuration Files

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

See also

- *SAP Mobile Server Configuration Files* on page 316
- *SAP Control Center Configuration Files* on page 321
- *Relay Server Configuration Files* on page 324
- *Agentry Configuration Files* on page 338

Cache Database Startup Options (cdboptions.ini) Initialization File

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

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

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

Table 45. Default Startup Options

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

See also

- *Changing Database Ports for SQLAnywhere Databases* on page 63
- *Changing SQLAnywhere Database Server Startup Options* on page 64

Agentry Configuration Files

Agentry Server configuration files are installed with SAP Mobile Server in *SMP_HOME*`\Servers\AgentryServer`.

See also

- *SAP Mobile Server Configuration Files* on page 316
- *SAP Control Center Configuration Files* on page 321
- *Relay Server Configuration Files* on page 324
- *Data Tier Configuration Files* on page 337

Agentry.ini Configuration File

The `Agentry.ini` file is the main Agentry Server configuration file. The installed version of this file is located in *SMP_HOME*`\Servers\AgentryServer`. For Agentry applications, this file is updated when you configure the Agentry Server instance using SAP Control Center, and is stored in *SMP_HOME*`\Servers\UnwiredServer\Repository\Agentry\default\`*agentryAppID*.

Agentry Server Configuration Sections

The following table lists the sections of configuration settings as listed within the SAP Control Center. It includes descriptions of their general purpose and areas of behavior affected by the settings contained in each section. Ultimately these settings are contained within the `Agentry.ini` configuration file for the Agentry Server. The preferred method of setting any configuration option is to use the SAP Control Center wherever possible. However, some settings may need to be initially set within the Agentry.ini file. In such cases these settings are noted in their descriptions, as well as in any procedures which describe modifying those settings.

Section	Description
Agentry	Contains settings related to the messages and events logs, as well as one to identify the Server instance.
Server	Contains settings for user, device, and expiration keys. These may be changed if new licenses are provided by Syclo in relation to changes in the usage of the platform. Also contains general sever configuration options related to time outs.
Configuration	This section contains settings for referencing other configuration files used by the Server. Also, the settings specifying if the Server is configured to provide debugging and whether the Server is for development or production uses.
Front Ends	This sections lists the dynamic link libraries (DLL's) to load for the supported Client-Server communications methods. This list is modified if different connection types are supported other than ANGEL.
ANGEL Front End	The configuration options in this section affect the behavior of communications between the Clients and Server when using the ANGEL connection type. It includes certificate file references, client and server authentication settings, cipher strengths for encryption, and time out values.
ANGEL Front End Ports	This section contains a list of one or more host names and ports upon which the Server will listen for requests from Clients. Multiple entries may be added here for environment support such as Network Address Translation, security and firewalls, and clustered environments.
Web Server Front End	This section contains configuration options that affect the behavior of communications between the Clients and Server when the Clients are web browsers. This includes both PC browsers and the RIM Blackberry Handheld Browser. Options related to authentication certificates, ports, and debugging for such communications are set here.

Section	Description
Web Server MIME Types	This section lists the MIME types related to Web Clients. The items listed here are in addition to those configured on the systems in use.
Server Side Clients	This section lists the DLL files used for Web Server communications. Currently only the entry for ServerSideClient.dll should be listed here. Other options within this section are reserved for future use.
System Connections	This section lists the DLL files to be loaded by the Server based on the system connections the application supports and uses. Each entry in this section must match a System Connection definition within the application. Additionally, a matching section must exists for the System Connection within the Agentry.ini file.
SQL-n	This section contains the configuration options for a single SQL Database system connection. Its settings control the behavior of this connection between the Server and a database system. There may be multiple SQL-n sections within the Agentry.ini file. One must exist for each SQL Database system connection definition within the application. The n portion of the section name must be replaced with the ID value generated by the Editor for the system connection definition.
Java-n	This section contains the configuration options for a single Java Virtual Machine system connection. Its settings control the behavior of this connection between the Server and a Java interface. There may be multiple Java-n sections within the agentyr.ini file. One must exist for each Java Virtual Machine system connection definition within the application. The n portion of the section name must be replaced with the ID value generated by the Editor for the system connection definition.
HTTPXML-n	This section contains the configuration options for a single HTTP-XML system connection. Its settings control the behavior of this connection between the Server and an HTTP server. There may be multiple HTTPXML-n sections within the Agentry.ini file. One must exist for each HTTP-XML system connection definition within the application. The n portion of the section name must be replaced with the ID value generated by the Editor for the system connection definition.
File-n	This section contains the configuration options for a single NT File system connection. Its settings control the behavior of this connection between the Server and the Server's host system. There may be only one File-n section within the Agentry.ini file. This section must exists for an NT File system connection definition within the application. The n portion of the section name must be replaced with the ID value generated by the Editor for the system connection definition.

Section	Description
Agents	This section is used by the Server for internal purposes. It should not be modified unless specifically instructed by a Syclo support or services associate.
SpinDoc	This section is used by the Server to determine the location of SQL script files. It should not be modified unless specifically instructed by a Syclo support or services associate.
LastUpdates	This section contains date and time values related to the last update of certain other sections within the Server's configuration files. It should never be modified and is used by the Server for internal purposes.

Agentry Server: [Agentry] Configuration Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the Agentry set of properties found in the SAP Control Center. These settings are also those found in the `[Agentry]` section of the `Agentry.ini` file.

Configuration Setting	Acceptable Values	Description
systemName	Any Text Value	This optional parameter specifies the name of the Agentry Server. This value is displayed in the title bar of the Server's GUI interface on the console. If no value is specified, or if the setting is omitted from the section, the default value of "Agentry Server" is displayed.
messageLogFile	Valid file name or path and file name. Default: messages.log	This parameter contains the name of the file to which message log items are written. In most cases the default of messages.log is sufficient. It can be changed to specify a different file name or location. The path can be relative to the Server's installation folder; or a full path including drive letter.
eventLogFile	Valid file name or path and file name. Default: events.log	This parameter contains the name of the file to which event log items are written. In most cases the default of events.log is sufficient. It can be changed to specify a different file name or location. The path can be relative to the Server's installation folder; or a fill path including drive letter.
logMaxSize	Numeric value, assumed to be in bytes. Default: 0	The maximum size of the messages.log file in bytes. A value of 0 indicates no maximum size. Any other value will result in the log file roll behavior when the messages.log file exceeds the value set here.

Configuration Setting	Acceptable Values	Description
logRollTime	Time of day in 24 hour clock format. Hour and minute of the hour only. Default: 1:45	The time of day, in 24 hour format (HH:MM) when the events and messages log files should be rolled. The Server must be running at the time specified here in order for the files to be rolled.
logRollCommand	Valid batch file name or path and batch file name. Default: roll.bat	The name of the batch file or command to roll the events and messages log files. The default roll.bat is installed with the Server. The command referenced here is executed when the log roll time (logRollTime) is reached, or the maximum log size (logMaxSize) is exceeded.
logRollsAtTime	Boolean value of "true" or "false." Default: true	Boolean value of true or false indicating whether to roll the messages and events log files based on the time in logRollTime.

Agentry Server: [Server] Configuration Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the Server set of properties found in the SAP Control Center. These settings are also those found in the `[Server]` section of the `Agentry.ini` file.

Configuration Setting	Acceptable Values	Description
userKey	Valid Syclo-issued user key	Initially this will contain the user key entered when the Server installer was run. It can be modified with an updated user key, issued by Syclo, if additional user licences are purchased.
deviceKey	Valid Syclo-issued device key	Initially this will contain the device key entered when the Server installer was run. It can be modified with an updated user key, issued by Syclo, if additional device licenses are purchased.
expKey	Valid Syclo-issued expiration key	Initially this contain the expiration key entered when the Server installer was run. It can be modified with an updated expiration key, issued by Syclo, if a new expiration date is set.

Configuration Setting	Acceptable Values	Description
autoRegister	Boolean of true or false. Default: true	This configuration option controls whether or not user ID's sent by the Client to the Server are added to the users.ini file automatically. If this option is set to false, the users.ini file will not be updated by the Server. Any user ID not listed in the users.ini file will not be allowed to log into the Server.
inactiveTimeout	Numerical value representing seconds. Default: 600	This is a duration value specifying how long the Server is to wait for response from a Client after the Server has sent it a message. If no response is received from the Client after this duration value, the user is logged out of the Server. This setting affects all Client communication types.
allowRelogin	Boolean of true or false. Default: true	This setting controls whether users can log into the Server from Clients with different IP addresses at the same time. If true, the original session for a user will be ended, with the user logged out, before they are logged in from the new IP address. If this option is false, the Server will reject any Client login attempts from a second IP address for a user currently logged in from a different address.

Note: Two additional options, notifyUserOnLogout and notifyUserOnShutdown may be listed in the [Server] section of the Agentry.ini file. These items are no longer supported and are scheduled to be deprecated in a future release of Agentry.

Agentry Server: [Configuration] Configuration Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the Configuration set of properties found in the SAP Control Center. These settings are also those found in the `[Configuration]` section of the `Agentry.ini` file.

Configuration Setting	Acceptable Values	Description
developmentServer	Boolean value of "true" or "false." Default: *See Description.*	This configuration option controls whether the Server behaves as a development or production server. This option is set by the installer based on the options selected there (development: true; production: false) and should not be changed accept at the instruction of a Syclo support or services associate.

Configuration Setting	Acceptable Values	Description
debug	Boolean value of "true" or "false." Default: *See Description.*	This configuration option controls whether certain log files are generated by the Server at run-time. It's default values are true for a development server and false for production. Log Files: Server.log, Server.user.log, AgentryJava.log.
localizations	Text value of valid locale names. Default: *none*	This configuration setting can contain one or more local names according to the ISO 639-1 standard. This option lists the supported languages, requiring corresponding override localization files, for an application. Each local name must be separated by a semi-colon.
localizationPath	Valid path relative to the Server's installation folder. Default: localizations	This option contains the location of localization files for use in the application. This path is relative to the Server's installation folder. If the default folder "localizations" does not exist, it must be created.
enableOverrideFile	Valid file name or path and file name. Default: Enables.ini	This configuration option specifies the name and location of the enables localization file. This setting can be changed when the name of this file is different from the default.
clientStringsFile	Valid file name or path and file name. Default: ClientText.ini	This configuration option specifies the name of the client strings localization file. This setting can be changed when the name of this file is different from the default.
applicationStringsFile	Valid file name or path and file name. Default: ApplicationText.ini	This configuration option specifies the name and location of the application strings localization file. This setting can be changed when the name of the file is different than the default.
applicationGlobalsFile	Valid file name or path and file name. Default: Globals.ini	This configuration option specifies the name and location of the globals localization file. This setting can be changed when the name of the file is different than the default.

Configuration Setting	Acceptable Values	Description
clientStringNames	Valid file name or path and file name. Default: ClientStringNames.ini	This configuration option specifies the name and location of the client string names localization file. This setting can be changed when the name of the file is different than the default.
overrideTypesFile	Valid file name or path and file name. Default: OverrideTypes.ini	This configuration option specifies the name and location of the override types localization file. This setting can be changed when the name of the file is different than the default.
transmitConfigurationFile	Valid file name or path and file name. Default: TransmitConfigurations.ini	This configuration option specifies the name and location of the transmit configurations override file. This setting can be changed when the name of the file is different than the default.
enableTransactionFailureHandling	Boolean value of “true” or “false.” Default: false	This configuration option enables or disables the transaction failure handling functionality within the Agentry platform. This option must be true in order for all aspects of this functionality to be enabled.

Configuration Setting	Acceptable Values	Description
enableFaieldTransactionLogging	Boolean value of "true" or "false." Default: true	This configuration option enables or disables the failed transaction que, which is a part of the transaction failure handling functionality. By default, when transaction failure handling is enabled, transactions that fail with a fatal error code result in the creation of a failed transaction que on the server containing the data from that transaction. If this option is set to false, this file is not generated. All remaining transaction failure handling behaviors remain the same, including the removal of the failed transaction from the client. NOTE: setting this option to false can result in lost data for fatal errors related to transaction processing. This option has no affect if transaction failure handling is disabled, i.e `enableTransactionFailureHandling` is false.
failedTransactionQueue	Valid folder name or path and folder name. Default: FailedTransactionsQueue	This configuration option specifies the name of the folder where the failed transaction queue files are stored. This path is relative to the installation folder of the Server. It may be changed if the default folder is not the desired location for these files.
failedTransactionFilenameFormat	Series of format strings and constants producing a valid XML file name.	This configuration option specifies the format of the file name for each failed transaction queue file generated by the Server. The format strings %{userid}, %{transaction_name}, %{date}, %{time}, and %{count} may be used. The file generated will always be in XML format and the file extension provided (.xml) should reflect this content type.

Agentry Server: [Front Ends] Configuration Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the Front Ends set of properties found in the SAP Control Center.

These settings are also those found in the `[Front Ends]` section of the `Agentry.ini` file. The items in this section are anonymous keys. Therefore, the following table lists the acceptable values, specifying library (DLL) files to be loaded by the Server upon startup in support of the different client-server connection types. Only those for front ends (connect types) in use should be listed in this section.

Library File Name	Description
AngelVine.dll	This file is listed in the [Front Ends] section by default for versions of Agentry beginning with 4.4. This file must be listed if one or more transmit configurations are defined to use the ANGEL connection type.
WebVine.dll	This file is listed in the [Front Ends] section when the Web Client is in use for an application implementation. When listed, the Server will take on certain web server-like behaviors, serving screens in HTML or WML format to web clients.

Agentry Server: [ANGEL Front End] Configuration Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the ANGEL Front End set of properties found in the SAP Control Center. These settings are also those found in the `[ANGEL Front End]` section of the `Agentry.ini` file. If the ANGEL connect type is not in use by at least one transmit configuration definition within the application the settings of this section will have no impact on the behavior of the Agentry Server.

Configuration Setting	Acceptable Values	Description
trustedCertificateStore	Valid Trusted Certificate Store File Name. Default: none	This option specifies the location of the trusted certificate store for the Server. This store is only used when Clients are required to provide authentication certificates during synchronization.
authenticationCertificateStore	Valid authentication certificate file. Default: AgentryServer.pfx	This option specifies the location of the authentication certificate for the Server. This file may be a Certificate file (.cer), Certificate store file (.sst), or Personal Information Exchange file (.pfx).

Configuration Setting	Acceptable Values	Description
authenticationCertificateStorePassword	Valid password for accessing the Server's authentication certificate. Default: syclo	This option specifies the password to access the Server's authentication certificate. The default password allows access to the default self-signed certificate file AgentryServer.pfx.
authenticationCertificateStorePasswordEncoded	Boolean value of "true" or "false." Default: false	This option specifies whether or not the certificate store password value is encoded in the Agentry.ini file. This option should not be changed manually. Rather, the quick password encoder utility will change it when run.
authenticateClient	Boolean value of "true" or "false." Default: false	This option controls whether or not Clients are required to provide an authentication certificate to the Server during synchronization. This certificate must be listed in the trusted certificate store in order for Clients to access the Server.
debug	Boolean value of "true" or "false." Default: false.	This configuration file controls whether or not the Server generates log files for communications using the ANGEL protocol. When true, the log file listed in debugFile is generated, as are individual log files for each Client during synchronization.
debugFile	Valid log file name or path name name. Default: ANGEL.log	This configuration option specifies the name of the log file generated by the Server when the debug option in this same section is true.
timeout	Duration value in seconds. Default: 300	This configuration option specifies how long the Server will keep a connection option with a Client when no messages or responses are received from the Client.

Configuration Setting	Acceptable Values	Description
keepAliveTime	Duration value in seconds. Default: 60	This configuration option specifies how often the Server will send a keep alive message to the Client when the Client is keeping an open connection (Push or Background Sending) with the Server. This message prevents the connection from expiring based on the timeout value.
minimumCipherStrength	Numeric value in bits. Default: 40. Configuration option commented out by default.	This configuration option specifies the minimum cipher strength for the encryption of data over a Client-Server connection. This configuration option is commented out by default, meaning the Server and Client will negotiate the maximum supported cipher strength by the client device. When set, devices unable of supporting the minimum set here will not be allowed to connect with the Server.
maximumCipherStrength	Numeric value in bits. Default: 512. Configuration option commented out by default.	This configuration option specifies the maximum cipher strength for the encryption of data over a Client-Server connection. This configuration option is commented out by default, meaning the Server and Client will negotiate the maximum supported cipher strength by the client device. When set, data will be encrypted using this maximum cipher strength.

Agentry Server: [Web Server Front End] Configuration Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the Agentry set of properties found in the SAP Control Center. These settings are also those found in the `[Web Server Front End]` section of the `Agentry.ini` file. The settings in this section apply only to Web Client functionality. They will have no affect on the Agentry Server's behavior if this client type is not in use.

Configuration Setting	Acceptable Values	Description
listenOn	Valid port number. Default: see description	This setting specifies the port number upon which the Server will listen for requests from web browser clients. The default for unsecured traffic is port 80 and 443 for SSL encrypted traffic. This may be changed to any valid and open port number. It should not be the same as any other port number used by the Server for other client-server communications.
permitGetCGI	Boolean value of “true” or “false.” Default: true	This setting specifies whether or not the Server should accept GET CGI requests from web browser clients.
debug	Boolean value of “true” or “false.” Default: false	This setting controls whether or not the log file for web client communications i generated.
xslDirectory	Valid folder or path. Default: xsl	This setting contains the name of the folder where the XSL files served to browser clients are stored. This is relative to the Server’s installation folder.
resourceDirectory	Valid folder or path. Default: resource	This setting contains the location of the folder where the resources for the web pages served to the browser clients are stored. This location is relative to the Server’s installation folder.
authenticationCertificateStore	Valid location of authentication certificate. Default: AgentryServer.pfx	This setting references the authentication certificate provided to browser clients by the Server. The default certificate file is a self-signed certificate provided by with the Server installation.

Configuration Setting	Acceptable Values	Description
authenticationCertificateStorePassword	Valid password for authentication certificate. Default: Syclo	This setting is the password value needed to access the authentication certificate referenced in authenticationCertificateStore. This value is changed when a different certificate is used that has a different password value.
authenticationCertificateStorePasswordEncoded	Boolean value of “true” or “false.” Default: false	This setting specifies whether or not the password referenced in authenticationCertificateStorePassword has been encoded by the quick password encoder utility. This setting should not be changed manually. It will be set to the proper value by the encoder utility when run.

Agentry Server: [Web Server MIME Types] Section

The `[Web Server MIME Types]` section of the `Agentry.ini` file is used to provide the proper Multipurpose Internet Mail Extensions (MIME) types settings for an application using the Web Server front end. The entries listed in this section are MIME types to be supported in addition to those registered on the host system of the Agentry Server. The entries for this section take the form:

```
fileExtension=MIMEType
```

If a MIME type is listed in the `[Web Server MIME Types]` section of the `Agentry.ini` file and also registered on the Windows host system of the Server, the entry in the `Agentry.ini` file will override the registered entry.

Agentry Server: [System Connections] Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the System Connections set of properties found in the SAP Control Center. These settings are also those found in the `[System Connections]` section of the `Agentry.ini` file. The items listed in this section are key and values, where the key portion is the same as the ID of the system connection definition for which the entry is added. Therefore, the following table lists the acceptable values, specifying library (DLL) files to be loaded by the Server upon startup in support of the different system connection types supported by the Agentry platform. This section should contain entries only for system connection in use by the application and defined within the application project. A

corresponding system connection type section (`[SQL-n]`, `[Java-n]`, `[HTTPXML-n]`, or `[File-n]`) must be added to the `Agentry.ini` file as well.

Note: The items in this section are typically added automatically by the Agentry Editor when needed and based on the system connections defined within the project when that project is published to the Agentry Server. Therefore it is rarely necessary to add these items manually. They are listed here for reference purposes only.

Library File Name	Description
ag3sqlbe.dll	This file is listed for each SQL Database system connection within the application.
ag3javabe.dll	This file is listed for each Java Virtual Machine system connection within the application.
ag3httpxml.dll	This file is listed for each HTTP-XML system connection within the application.
ag3filebe.dll	This file is listed when an NT File System connection is in use within the application.

Agentry Server: [SQL-n] Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the SQL-*n* set of properties found in the SAP Control Center. These settings are also those found in the `[SQL-n]` section of the `Agentry.ini` file. When adding such a section the *n* portion of the section name must match the ID of the corresponding SQL Database system connection definition in the application. If no such system connection exists, the settings in this section will not affect the behavior of the Agentry Server or the application.

Configuration Setting	Acceptable Values	Description
name	Text value. Default: *null*	This setting can contain any text value used to identify the system connection in log files and other areas. This should be set to a unique value especially when working with multiple system connections.

Configuration Setting	Acceptable Values	Description
sharedConnections	Zero or positive integer value. Default: 0	This setting specifies the number of connections created by the Server upon startup. If greater than 0, a connection pool is created and data from a single Client may be processed on any of these connections, including multiple different connections during a single transmission. If set to 0, each Client transmission will use a single database connection from start to finish.
queryInitFile	Valid file or path and file name to properly formatted query initialization file. Default: sqlBE.ini	This setting specifies the name and location for the query initialization file used by this system connection. This initialization file contains numerous sections related to various Server events when working with a database system connection. The file referenced here is expected to contain all of these sections in the proper format.
debug	Boolean value of "true" or "false." Default: *see description*.	This setting specifies whether or not the log file referenced in queryDebugFile is generated by the Server. It also enables or disables the creation of the sql.query.user.log file. This setting is false by default for production servers and true for development servers.
queryDebugFile	Valid file or path and file name. Default: sql.query.log	This setting specifies the log file generated by the server to contain log messages for the system connection. This includes query results and database generated messages. This file name may be changed when multiple database system connections are in use. The name may include the token %1 to include the text value from the name configuration option in the log file's name. Example: "%1.sql.query.log" would produce a log file named similar to: "MyDB.sql.query.log"

Configuration Setting	Acceptable Values	Description
saveQuery	Boolean value of "true" or "false." Default: *see description*	This setting specifies whether or not the query being executed is included in the query debug log file. This setting is false by default for production servers and true for development servers.
saveResult	Boolean value of "true" or "false." Default: *see description*	This setting specifies whether or no the resulting messaging from the database is included in the query debug log file. NOTE: Error messages from the database are always written to the log file. This setting is false by default for production servers and true for development servers.
queryConstantsFile	Valid file name or path and file name. Default: *see description*	This setting specifies the file name and location, relative to the Server, of the query constants file. The default for this setting depends on the type of database selected during installation and will be set to either oracle_sd.ini or sqlserver_sd.ini.
preloadQueryInitFile	Boolean value of "true" or "false." Default: *see description*	This setting specifies whether query initialization file named in the queryInitFile setting is loaded when the Server starts up, or if it is read by the Server for each new Client connection. If false, changes to the query initialization file will not take affect until the Server is restarted. This setting is false by default for production servers and true for development servers.
enableAuthentication	Boolean value of "true" or "false." Default: true	This value specifies whether or not users are authenticated against the back end system for this system connection. At least one system connection within the application must perform user authentication.
enablePreviousUserAuthentication	Boolean value of "true" or "false." Default: true	This value specifies whether or not previous users are authenticated against the back end system for this system connection. This authentication occurs when a user change occurs on the Client.

Configuration Setting	Acceptable Values	Description
useUserDBAuthConnection	Boolean value of "true" or "false." Default: false	This value specifies whether or not the connection used to authenticate the user should be the same one used to process synchronization for that user. If false, the Server will use a different connection.
allowUserToChangePassword	Boolean value of "true" or "false." Default: true	This value specifies whether or not users can change their passwords on the Client when the database indicates the password has or is about to expire. If enableAuthentication is false, this setting has no affect.
dbConnectionName	Text value with valid system connection name. Default: *set by installer*.	The value of this setting contains the TNS name or ODBC DSN name the Server will use to connect with the database.
dbConnectionUserID	Text value with valid user ID	The user ID the Server will use to log into the database.
dbConnectionPassword	Text value with valid password. Default: *set by installer*.	The password for the user ID the Server will use to log into the database.
dbConnectionPasswordEncoded	Boolean value of "true" or "false." Default: false	Specifies whether or not the value in dbConnectionPassword has been encoded by the quick password encoder utility. This value should not be changed manually as the encoder utility will set it appropriately.
dbConnectionType	"Oracle" or "ODBC." Default: *set by installer*.	Specifies whether the Server is connecting to the database using an ODBC or Oracle Net Service Name.
disconnectFromDB	Boolean value of "true" or "false." Default: false	This Boolean value specifies whether or not the Server should disconnection from the database at the time specified in disconnectTime. This is set to true when the database performs periodic batch or other processing and when such processing dictates that no connections should be made to the database system.

Configuration Setting	Acceptable Values	Description
disconnectTime	24 hour clock value in hours and minutes. Default: 2:30	This setting contains the time of day when the Server should disconnect from the database. This setting has no affect if disconnectFromDB is false.
reconnectTime	24 hour click value in hours and minutes. Default: 4:00	This setting contains the time of day when the Server should reconnect to the database. It must be later than the time specified in disconnectTime. This setting has no affect if disconnectFromDB is false.
timeZoneName	Valid time zone name.	This setting contains the name of the time zone to which date and time values are converted when received from Clients; or from which they are converted when sent to Clients. This conversion is based on the application definitions. This setting is only used to specify a time zone other than the one for the back end system.

Agentry Server: [Java-n] Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the Java-*n* set of properties found in the SAP Control Center. These settings are also those found in the `[Java-n]` section of the `Agentry.ini` file. When adding such a section the `n` portion of the section name must match the ID of the corresponding Java Virtual Machine system connection definition in the application. If no such system connection exists, the settings in this section will not affect the behavior of the Server or the application.

Configuration Setting	Acceptable Values	Description
enableAuthentication	Boolean value of “true” or “false.” Default: true	This value specifies whether or not users are authenticated against the back end system for this system connection. At least one system connection within the application must perform user authentication.
enablePreviousUserAuthentication	Boolean value of “true” or “false.” Default: true.	This value specifies whether or not previous users are authenticated against the back end system for this system connection. This authentication occurs when a user change occurs on the Client.

Configuration Setting	Acceptable Values	Description
name	Text value	This setting can contain any text value used to identify the system connection in log files and other areas. This should be set to a unique value especially when working with multiple system connections.
classPath	Valid fully qualified paths.	This setting contains multiple path values specifying the location of different Java resources. This can include .jar files used by the application. Each path is required to contain be a fully qualified path. Each entry must be separated by a semi-colon. All paths for this setting are added to the system variable CLASSPATH of the Server's host system.
serverClass	Valid Java Server Class extension	This option is set to specify the application-specific extension of the AJ-API syclo.agentry.server class. This setting is set when this API class has been extended by a server class for the application. If no such class exists, the default syclo.agentry.server class is assumed and should not be set for this configuration option.
outputDirectory	Valid path value, relative to the Server.	This configuration option is set to the location where the compiled .class files produced by the JVM are to be stored for execution. The standard is to use the sub-folder Java from the Server's installation folder. However, another location may be used. Whichever location is used it must be specified here. This same path value must also be a part of the system's CLASSPATH. This can be accomplished by setting that system variable, or by adding the value to the classPath configuration option.

Configuration Setting	Acceptable Values	Description
sourceDirectory	Valid path value, relative to the Server.	This configuration option specifies the location of the .java files for the Steplet, ComplexTable and DataTable classes. This location is where these files are placed prior to compilation by the JVM. Note that this is not the location where the files reside in any project for Java files related to the application. The .java files are written to this location by the Server at run-time. Their contents are based on the definitions in which they are contained.
deleteSource	Boolean value of “true” or “false.” Default: false	This configuration option controls whether the files stored in sourceDirectory are deleted after successfully compiled by the JVM. This may be necessary in a development environment if the Java source files are modified outside of the Editor.
printStackTrace	Boolean value of “true” or “false.” Default: false	Setting this option to true will result in messages generated by the exception stack trace being printed to the events.log file produced by the Server.
printBusinessLogicStackTrace	Boolean value of “true” or “false.” Default: false.	Setting this option to true will result in messages generated by the AJ-API exception class JavaBusinessLogicError being printed to the events.log file produced by the Server.
timeZoneName	Valid time zone name. Default: null	This setting contains the name of the time zone to which date and time values are converted when received from Clients; or from which they are converted when sent to Clients. This conversion is based on the application definitions. This setting is only used to specify a time zone other than the one for the back end system.
initialHeapSize	Positive integer value in megabytes.	This setting specifies the initial heap size allocated by the JVM when it is started by the Server. This is equivalent to passing the -Xms command-line option to the JVM.

Configuration Setting	Acceptable Values	Description
maxHeapSize	Positive integer value in mega-bytes.	This setting specifies the max heap size the JVM will allocate when running. This is the equivalent to passing the -Xmx command-line option to the JVM
reduceOSSignalUse	Boolean value of “true” or “false.” Default: false	This option, when set to true, will result in the JVM not being shut down when the Server shuts down. This is only applicable when the Server is running as Windows service.
nonStandardJavaOptions	Text value.	This option can contain additional command-line options passed to the JVM when started by the Server. Any valid options to the JVM may be listed here as they would appear on the command-line. If either -Xms or -Xmx are specified here, the corresponding configuration settings should not be used and should be omitted from the [Java-n] section.
lowMemoryPercentage	Positive integer value as a percentage.	This configuration option can contain a numeric value treated as a percentage. If available system memory falls below this value a message is written to the events.log file generated by the Server.
performCompile	Boolean value of “true” or “false.” Default: true.	This configuration option specifies whether or not to compile the .java files into .class files for the Steplet, ComplexTable, and DataTable classes written for the application. Setting this to false may increase performance in a production environment. In this case, the last built .class files will be reused each time such a class is called within the application. If set to true, each class of the above types will be recompiled prior to execution for each user.
debugFile	Valid file name or path and file name relative to the Server.	This option specifies the name of the log file generated by the Server with messages produced by the AJ-API. The token %1 may be included to make add the value in the “name” configuration option of the [Java-n] section to the file name. Example: “%1.java.log” will produce a a file name similar to: “MyJVM.java.log”

Configuration Setting	Acceptable Values	Description
debug	Boolean value of “true” or “false.” Default: *see description*	This option specifies whether debugging messages are generated by the Server in relation to the JVM System Connection for the [Java-n] section. Note that this option need not be set to true in order to have the JVM produce its own debugging information via the nonStandardJavaOptions setting. The debug option is set by default to true for development servers and false for production servers.

Agentry Server: [HTTPXML-n] Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the HTTPXML-*n* set of properties found in the SAP Control Center. These settings are also those found in the `[HTTPXML-n]` section of the `Agentry.ini` file. When adding such a section the *n* portion of the section name must match the ID of the corresponding HTTP-XML system connection definition in the application. If no such system connection exists, the settings in this section will not affect the behavior of the Server or the application.

Configuration Settings	Acceptable Values	Description
name	Text Value	This setting can contain any text value used to identify the system connection in log files and other areas. This should be set to a unique value especially when working with multiple system connections.
baseURL	Valid URL and optional Port number. Default: *none*	This is the base URLaddress of HTTP requests made by the Server for this system connection. This option must be set prior to starting the Server.
xmlNamespaces	Valid XML names paces within XML schema	This configuration option can contain one or more XML name spaces within the XML schema. Each name space entry for this option must be separated by a pipe (|) symbol.
debugFile	Valid file name or path and file name relative to Server folder. Default: httpxml.log	This option specifies the name of the log file generated by the Server. The token %1 may be included to add the value in the “name” configuration option of the [HTTPXML-n] section to the file name. Example: “%1.httpxml.log” will produce a a file name similar to: “MyXML.httpxml.log”

Configuration Settings	Acceptable Values	Description
user Debug File	Valid file name or path and file name relative to Server folder. Default: httpxml.%1.log	This option specifies the name of the log file generated by the Server. The token %1 may be included to add the value in the "name" configuration option of the [Java-n] section to the file name. The token %2 may be added to include the user ID for the Client in the name. Example: "%1.%2.httpxml.log" will produce a a file name similar to: "MyJVM.Smith.httpxml.log"
constantsFile	Valid file name or path and file name to constants file. Default: httpxml_sd.ini	This option names the constants file used for this system connection. This file contains constant named values that may be referenced by the HTTP-XML components of the application.
debug	Boolean value of "true" or "false." Default: *see description.*	This option specifies whether debugging messages are generated by the Server in relation to the HTTP-XML System Connection for the [HTTPXML-n] section. This option is set by default to true for development servers and false for production servers.
httpConnectTimeout	Positive integer treated as milliseconds	This option specifies the time out period in milliseconds for this system connection. If this time out value is exceeded the connection will be closed. This timer resets when data is exchanged between Server and the back end system during Client synchronization.
enableAuthentication	Boolean value of "true" or "false." Default: true	This value specifies whether or not users are authenticated against the back end system for this system connection. At least one system connection within the application must perform user authentication.
xmlValidateOnParse	Boolean value of "true" or "false." Default: false	This option specifies whether or not to validate the XML against the known schema when returned after a request. If false, this validation is not made.
saveRequest	Boolean value of "true" or "false." Default: *see description*	This option specifies whether or not to write the request made to the back end system to the log file for the system connection. The defaults for this option are true for a development server and false for a production server.

Configuration Settings	Acceptable Values	Description
saveResult	Boolean value of "true" or "false." Default: *see description*	This options specifies whether or not to write the resulting messaging of a request to the log file for the system connection. The defaults for this option are true for a development server and false for a production server.

Agentry Server: [File-n] Section

The following table lists the available configuration options, acceptable values, and descriptions of the settings in the File-*n* set of properties found in the SAP Control Center. These settings are also those found in the `[File-n]` section of the `Agentry.ini` file. When adding such a section the `n` portion of the section name must match the ID of the corresponding NT File System connection definition in the application. If no such system connection exists, the settings in this section will not affect the behavior of the Server or the application.

Configuration Setting	Acceptable Values	Description
enableAuthentication	Boolean value of "true" or "false." Default: false	This value specifies whether or not users are authenticated against the host system for this system connection. At least one system connection within the application must perform user authentication.
enablePreviousUserAuthentication	Boolean value of "true" or "false." Default: false	This value specifies whether or not previous users are authenticated against the host system for this system connection. This authentication occurs when a user change occurs on the Client. If allowPreviousUserLogin is false, this setting has no affect.
userDomain	Text value of a configured user domain.	This setting can contain the domain under which all users will be logged into the host system
tempPath	Valid full path.	This setting specifies the location where temporary files produced by the Server for file system processing are stored. This is normally the batch files defined within the application. If this option is null or not listed in the section the configured Windows temporary folder is used.

Configuration Setting	Acceptable Values	Description
allowPreviousUserLogin	Boolean value of “true” or “false.” Default: true	This setting controls whether or not previous users are logged into the file system when a user change occurs on the Client. If not logged in, no file system connection processing defined in the application is performed for the previous user.
name	Text value.	This setting can contain any text value used to identify the system connection in log files and other areas. This should be set to a unique value especially when working with multiple system connections.
debug	Boolean value of “true” or “false.” Default: *see description.*	This option controls whether or not log file information is generated by the Server for the system connection. The defaults for this option are false for development servers and true for production servers.

Other Agentry Configuration Files

The following configuration files must be manually edited.

Agentry Server SqlBe.ini Configuration File Sections

The following table lists the sections available within the `SqlBe.ini` configuration file for the Agentry Server. It includes descriptions of their general purpose and areas of behavior affected by the settings contained in each section. Note that this configuration file is specific to SQL Database system connections. Its contents will not have any impact on applications not using such a connection. It will also not impact any other system connection types. Finally, multiple files with different names may exist for a single Agentry Server, if that Agentry Server has multiple SQL Database system connections. The file `SqlBe.ini` is the default name of this file. It is referenced for a particular system connection in the `[SQL-n]` section of the `Agentry.ini` file in the configuration setting `queryInitializationFile`. If it is necessary to use multiple files for multiple SQL Database system connections, the `queryInitializationFile` setting should be modified for each system connection section to reflect the name of the file for that section.

Each section of this file represents a event of the Agentry Server related to its communications with the database. Each section can contain a list of one or more SQL scripts to be executed against the database by the Agentry Server when the corresponding event occurs. Any given section may be empty if no special processing is necessary for the event.

Section Name	Description
DbConnect	The queries listed in this section are run when the Server starts up and opens a connection to the database.
DbDisconnect	The queries listed in this section are run when the Server closes its connection to the database just before shutdown.
DbConnectionInit	The queries listed in this section are whenever the Server opens a additional connections to the database during client-server synchronization.
ValidateUser	These queries are run whenever a Client logs into the Server for the purpose of validating that user. At least one of the scripts listed in this section is expected to return data indicating the results of the validation attempt.
ValidatePreviou-sUser	These queries are run to validate a previous user against the database. This occurs when a user change takes place on the Client. At least one of the scripts in this section is expected to return data indicating the results of the validation attempt.
LoggedIn	Any queries listed in this section are run after a Client has successfully logged into the Server and the ValidateUser scripts inidcate the user has been valida-ted.
LoggedOut	Queries in this section are run just prior to the user being logged out of the Server.
UserInfo	Any queries listed in this section are run after those in LoggedIn. The purpose of the UserInfo queries is to create data values referencable in the <<user.in-fo>> data tag and are available to any subsequent queries for that user. This includes those in the SqlBe.ini file as well as those contained in the application definitions.
ApplicationGlobals	This section contains queires expected to return values to override the values of one or more Global definitions within the application. The Globals overriden with these scripts should be those pertaining to general application behavior.
UserGlobals	This section contains queries expected to return values to override the values of one or more Global definitions within the application. The Globals overriden with these scripts should be those pertaining to individual users or user groups.
LoginFailed	This section contains the queries to run when those queries in the ValidateUser section indicate the user has failed validation for any reason.
LoginBlocked	This section contains the queires to run when those queries in the ValidateUser section indicate the user has been blocked from access to the back end.

Section Name	Description
ChangePassword	These queries are run to handle a change of password for the user. These scripts should handle the necessary processing to perform the actual password change within the databasae.
ChangePassword-Failed	These queries are run if the scripts listed in ChangePassword return a failure. They may return messaging, displayed to the user on the Client, indicating the reason for the failure.
EnablePush	These queries are run whenever a Client logs in to the Server using a transmit configuration defined to allow push data to be received by the Client.
DisablePush	These queries are run whenever a Client disconnects from the Server after having been connected to receive push data.
TimeZone	These queries can be run to return to the time zone values used to convert date and time values recieved from or sent to the Client. This time zone indicates the one for the database system.
Misc	This section contains queires run by the Server for specific and internally determined situations. It is listed here for the sake of completeness, as it does appear in the SqlBe.ini file. However, neither this section nor the scripts listed within it should be modified except by Syclo support or services associates, or at their direction.

Override File Format: Enables.ini

Shown here is an example of the contents of the `Enables-` and `EnablesBase.ini` files. This is then followed by a description of this format.

```
[Customers]

screencolumn.ShowCustomers.ShowCustomers_List_PPC.CustomerID=true

screencolumn.ShowCustomers.ShowCustomers_List_PPC.CompanyName=true

action.AddCustomer=true

action.EditCustomer=true
```

The contents of this file are application specific and therefore will differ from one application to the next. The above is an example for a mobile application built for the Northwind demonstration database in SQL Server.

This override file contains one section for each module defined within the application, named `[`*`ModuleName`*`]`.

The following line :

```
screencolumn.ShowCustomers.ShowCustomers_List_PPC.CustomerID=true
```

represents the screen column (`screencolumn`) `CustomerID` in the `ShowCustomers_List_PPC` list screen, which is a child definition to the `ShowCustomers` screen set. Each override item's key follows a defined format:

```
defType.ModuleLevelParent.parent.parent.definitionName
```

The following items are the explanations to each component of the key

- `defType` - The type of definition being overridden by the item. Examples include:
  - `action`
  - `screencolumn`
  - `field`
- `ModuleLevelParent` - This is the name of the module-level parent definition of the definition whose enabled state is being overridden.
- `Parent.Parent`.. - The name of each subsequent ancestor definition of the definition being overridden is a part of the key.
- `definitionName` - The name of the definition being overridden.

Monitoring Database Schema

The monitoring database includes several tables from which information is retrieved by SAP Control Center.

These system tables are accessible to any user. The contents of monitoring tables can be changed only by SAP Mobile Platform components.

You can browse the contents of these tables by using any database browsing tool.

See also

- *Configuring Monitoring Performance Properties* on page 132

mms_rbs_request Table

Detailed history information for replication-based synchronization.

Column name	Column type	Description
id	integer	The identifier of the data row.
summaryId	varchar(50)	The identifier for the row summary information.

Column name	Column type	Description
deviceId	varchar(255)	The identifier of the physical device performing the synchronization request.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the MBO data synchronization or operation replay activity.
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
startTime	timestamp	The date and time the synchronization request was initiated.
endTime	timestamp	The date and time the synchronization request was completed.
syncTime	integer	The total time of the synchronization or operation replay activity, in milliseconds.
sendRows	integer	The number of rows downloaded during the mobile business object (MBO) synchronization. If 1 appears, the action was an operation replay.
isError	bit	Whether an error has occured during the synchronization or replay: 1 if errors were recorded, 0 if no errors were recorded.
sentBytes	integer	The number of bytes downloaded in the transaction.
receivedBytes	integer	The number of bytes uploaded in the transaction.
syncPhase	varchar(20)	The current synchronization activity: upload or download. During upload, a client initiates operation replays to execute MBO operations on the back-end system. During download, a client synchronizes with SAP Mobile Server to receive the latest changes to an MBO from the back-end system.
mboNames	varchar(500)	The MBO that downloaded information.
operationNames	varchar(500)	The operation replay.

Column name	Column type	Description
operationReplays	integer	The number of operation replays performed. Zero (0) indicates that information was downloaded by the MBO during a synchronization action.
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
applicationId	varchar(100)	The identifier for the application information.

mms_rbs_request_summary Table

Summary history information for replication-based synchronization transactions.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device performing the synchronization request.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO) data synchronization or operation replay activity.
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
startTime	timestamp	The date and time the synchronization request was initiated.
endTime	timestamp	The date and time the synchronization request was completed.
request syncTime	integer	The total time of the synchronization or operation replay activity, in milliseconds.
totalReceivedRows	integer	Always 1.
totalErrors	integer	The number of all exceptions during the synchronization request.

Column name	Column type	Description
totalsentBytes	integer	The number of all bytes dowloaded by the MBO.
totalreceivedBytes	integer	The number of all bytes uploaded by the MBO.
totalOperationReplays	integer	The number of all operation replays for the MBO.
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
applicationId	varchar(100)	The identifier for the application information.
mbo	varchar(500)	The MBO that downloaded information.

mms_rbs_mbo_sync_info Table

A subset of the mms_rbs_request table data.

Column name	Column type	Description
id	integer	The identifier of the data row.
packageName	varchar(255)	The package name of the mobile business object (MBO) data synchronization.
domain	varchar(255)	The domain to which the package involved in synchronization belongs.
mboName	varchar(255)	The name of the MBO performing the transaction.
startTime	timestamp	The date and time the synchronization request was initiated.
endTime	timestamp	The date and time the synchronization request was completed.
syncTime	integer	The total time of the synchronization, in milliseconds.

Column name	Column type	Description
isError	bit	Whether errors have occured during the synchronization: 1 if errors were recorded, 0 if no errors were recorded.

mms_rbs_operation_replay Table

A subset of the mms_rbs_request table data.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device performing the operation replay.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO).
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
startTime	timestamp	The date and time the operation replay request was initiated.
endTime	timestamp	The date and time the operation replay was completed.
processTime	integer	The total time of the operation replay, in milliseconds.
mbo	varchar(255)	The MBO performing the transaction.
operation	varchar(255)	The operation performing the operation replay.
isError	bit	Whether errors occurred during the synchronization or replay: 1 if errors were recorded, 0 if no errors were recorded.

Column name	Column type	Description
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
applicationId	varchar(100)	The identifier for the application information.

mms_mbs_message Table

Detailed history information for message-based synchronization.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device performing the operation replay.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO).
domain	varchar(255)	The domain to which the package involved in synchronization or operation replay belongs.
receiveTime	timestamp	The received time of inbound message. Not applicable to outbound messages.
pushTime	timestamp	The pushed time of outbound message. Not applicable to inbound messages.
startTime	timestamp	The date and time the message was initiated.
endTime	timestamp	The date and time the message was completed.

Column name	Column type	Description
processTime	integer	The total time of the message, in milliseconds.
mboName	varchar(255)	The MBO performing the message.
operationName	varchar(255)	The operation performing the message.
messageType	varchar(50)	The type of message. One of: SUBSCRIBE, UNSUBSCRIBE, OPERATION_REPLAY, RECOVER, SUSPEND, RESUME, RESUMENOREPLAY, IMPORT_DATA, DATA_RESET, LOGIN, UNKNOWN_TYPE.
isError	bit	Value is 1 if errors were recorded during transaction. 0 if no errors recorded.
payloadSize	integer	The size of the message payload.
isPushMsg	bit	Value is 1 if the message is outbound, 1 if otherwise.
isRequestMsg	bit	Value is 1 if the message is inbound, 0 if otherwise.
isSubscription	bit	Value is 1 if the message is a subscription request, 0 if not.
isOperationReplay	bit	Value is 1 if the message is a message-based operation replay, 0 if not.
sentPayloadSize	integer	The payload size of the outbound message.
receivedPayloadSize	integer	The payload size of the inbound message.
isMonitored	bit	Value is 1 if MBO is monitored, 0 if not.

Column name	Column type	Description
isLogged	bit	Value is 1 if the domain is logging data, 0 if not.
applicationId	varchar(100)	The identifier for the application information.

mms_security_access Table

Information about security and user access.

Column	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device used during the authentication request.
userName	varchar(255)	The name of the user requesting authentication.
packageName	varchar(255)	The package name of the authentication request.
domain	varchar(255)	The domain to which the package belongs.
securityConfiguration	varchar(255)	The name of the security configuration performing the authentication.
access_time	timestamp	The time the request for access was made.
outcome	bit	The outcome of the authentication request: 1 means authentication passed; 0 means authentication failed.
reason	longvarchar	Reason for authentication failure.
applicationId	varchar(100)	The identifier for the application information.

mms_rbs_outbound_notification Table

Outbound notification information for replication-based synchronization packages.

Column name	Column type	Description
id	integer	The identifier of the data row.
deviceId	varchar(255)	The identifier of the physical device receiving the outbound notification.
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the MBO.
domain	varchar(255)	The domain to which the package belongs.
publicationName	varchar(255)	The synchronization group of the outbound notification.
notificationTime	timestamp	The time the outbound notification was sent.
subscriptionId	integer	The identifier for the subscription.
subcriptionEnabled	bit	Whether the subscription is enabled. If 1, it is. If 0, it is not.
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
applicationId	varchar(100)	The identifier for the application information.

mms_data_change_notification Table

Information about data change notifications (DCNs) for messaging-based synchronization.

Column name	Column type	Description
packageName	varchar(255)	The package name of the mobile business object (MBO) affected by the DCN.
domain	varchar(255)	The domain to which the package involved in DCN belongs.
publicationName	varchar(255)	The synchronization group of the DCN.
notificationTime	timestamp	The time the DCN was sent.
processTime	integer	The total time to process the DCN, in milliseconds.
affectedRows	integer	The rows affected by the DCN.
isMonitored	bit	Whether the MBO is monitored. If 1, it is monitored. If 0, it is not.
isLogged	bit	Whether the domain is logging data. If 1, domain is logging data. If 0, it is not.
mboName	varchar(500)	The MBO that downloaded information.

mms_concurrent_user_info Table

Information about concurrent users.

Column name	Column type	Description
userName	varchar(255)	The name of the user associated with the device ID.
packageName	varchar(255)	The package name of the mobile business object (MBO) affected by the data change notification (DCN).
curTime	timestamp	The current time.

Column name	Column type	Description
domain	varchar(255)	The domain to which the package involved in DCN belongs.
type	bit	The type of request. If 1, it is a messaging request. If 0, it is a replication request.

mms_queue_info Table

Information about the messaging queue.

Colum	Column type	Description
queueName	varchar(255)	The name of the queue.
pendingItem	integer	The number of pendings messages in the server queue.
curTime	timestamp	The current time.

mms_sampling_time Table

Information about the sampling time.

Column	Column type	Description
sampling_time	timestamp	The sampling time
id	integer	The unique key of the table

cache_history Table

Saves the history events on the SAP Mobile Server cache by a deployed package.

Column	Column type	Description
package_name	varchar(128)	The package name of the mobile business object.
activity_type	integer	The event by activity type: • 1=ON DEMAND FULL REFRESH • 2=ON DEMAND PARTITIONED REFRESH • 3=CACHE QUERY

Column	Column type	Description
cache_name	varchar(128)	The SAP Mobile Server cache name.
mbo_name	varchar(128)	The mobile business object that triggered the activity.
start_time	datetime	The recorded start date and time of the activity.
duration	bigint	The recorded duration of the activity.
partition_key	varchar(128)	The partition key value.
host_name	varchar(64)	The host name.
process_id	varchar(64)	The process id.
unique_row_id	numeric(10,0)	Internal identifier only.

cache_history Stored Procedures

The cache_history table uses several stored procedures.

Procedure	Parameters	Result
`get_lastfullrefresh`	• packagename varchar(128) • cachename varchar(128)	The last full refresh value.
`get_lastinvalidatetime`	• packagename varchar(128) • cachename varchar(128)	The last invalid date value.
`get_lastupdatetime`	• packagename varchar(128) • cachename varchar(128)	The last update value.

cache_statistic Table

Saves the SAP Mobile Server cache status details.

Column	Column type	Description
package_name	varchar(128)	The package name.
cache_name	varchar(128)	The cache name.

Column	Column type	Description
last_full_refresh	datetime	The last date and time a full cache refresh occurred.
last_update	datetime	The last date and time a cache update occurred.
last_invalidate	datetime	The last date and time an invalidate cache occurred.

cache_statistics Stored Procedures

Provide aggregations of cache activities over a date range for a mobile business object.

Procedure	Parameters	Result
`get_package_mbo_maxfullrefreshtime` `get_package_mbo_minfullrefreshtime`	• @packagename varchar(128) • mboname varchar (128) • startdate datetime • endate datetime	The minimum or maximum value of the duration for the mobile business object over the start date and end date range for a refresh activity.
`get_package_mbo_maxcachewaittime` `get_package_mbo_mincachewaittime`	• packagename varchar(128) • mboname varchar(128) • startdate datetime • endate datetime	The minimum or maximum value of the duration for the mobile business object over the start date and end date range for a cache activity.
`get_package_mbo_averageeachewaittime` `get_package_mbo_averagefullrefereshtime`	• packagename varchar(128) • cachename varchar(128) • startdate datetime • endate datetime	The average value of the duration for the mobile business object over the start date and end date range for a cache or a refresh activity.

Procedure	Parameters	Result
`get_package_mbo_access_count` `get_package_mbo_ondemandrefresh_count`	• packagename varchar(128) • cachename varchar(128) • startdate datetime • endate datetime	The count of access and on demand refresh activities for mobile businss object over the start date and end date range.

Domain Log Categories

Domain log data provides detailed statistics for all aspects of device, user, application, domain, and data synchronization related activities.

Synchronization Log

Synchronization logs include data related to different aspects of data synchronization, including data, subscriptions, operations, result checker, cache refresh and the data service and SAP Mobile Server interface. Using data in these logs and the correlation tool, you can follow the data path between the enterprise information system (EIS), SAP Mobile Server, cache database, and user application connection.

To find out about	See
Data synchronization transactions	Data Sync statistics
Data services requests made to the Enterprise information system (EIS)	DS Interface statistics
Cache database (CDB) activities	Cache refresh statistics
EIS error codes or failures resulting from Mobile Business Object operations against the EIS datasource	Result Checker statistics (coding required)
Moving MBO operations from a mobile device to the CDB	Operation replay statistics
Moving data between a mobile device and the CDB	Subscription statistics

Data Sync

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

Data Sync – basic statistics for individual data synchronizations:

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Sync Group – the synchronization group associated with the request.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Operation Replay

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

Operation Replay – statistics for moving MBO operations (typically create, update, and delete) from the device cache to the cache database cache on SAP Mobile Server:

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.

- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Subscription

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

Subscription – statistics for transferring data between mobile devices and the cache database on SAP Mobile Server:

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- Subscription Type – the type of subscription used, including SUBSCRIBE, UNSUBSCRIBE, RECOVER, SUSPEND, and RESUME.
- Subscription ID – the identifier associated with the subscription.
- Sync Group – the synchronization group associated with the request.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Result Checker

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

Result Checker – EIS error codes or failures resulting from Mobile Business Object operations against the EIS datasource (requires coding):

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- Class – the class used for the result checker.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- None

Cache Refresh

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

Cache Refresh – statistics for cache database activities:

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Cache Group – the cache group name.
- CacheRow Count – the number of cached rows.

- EIS Row Count – the number of rows retrieved from the enterprise information system (EIS).
- Insert Count – the number of rows inserted in the cache.
- Update Count – the number of rows updated in the cache.
- Delete Count – the number of rows deleted from the cache.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Refresh Type
- Virtual Table Name
- Partition Key
- Pre Update Cache Image (payload)
- Post Update Cache Image (payload)

DS Interface

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

DS Interface – statistics for data services requests made to the Enterprise information system (EIS):

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.

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

Note: Additional detail columns:

- Operation Type
- Virtual Table Name
- Input Attributes (payload)
- Input Parameters (payload)

Device Notification Log

Device notification logs include logging data for server-initiated synchronization notifications between SAP Mobile Server and devices.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Sync Group – the synchronization group associated with the request.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Data Change Notification Log

Data Change Notification (DCN) logs include logging data for data change notifications between an enterprise information system (EIS) and an MBO package, for general and Hybrid App DCN.

General Data Change Notification

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

- Time – the time and date stamp for the log entry.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

- Payload

Hybrid App Data Change Notification

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

- Time – the time and date stamp for the log entry.
- Hybrid App ID – the unique identifier associated with a Hybrid App.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Package – the name of the package to which the subscription belongs.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Operation – the MBO operation.
- Subject – the Hybrid App DCN request subject line.
- From – the "From" value for the Hybrid App DCN request.
- To – the "To" value for the Hybrid App DCN request.
- Body – the message body for the Hybrid App DCN request.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.

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

Note: Additional detail columns:

- Payload

Security Log

Security logs provide security details for individual applications, application connections, and users. Logs capture authentication failures and errors, and provide supporting information that identifies request-response messaging, package and MBO details, security configuration, and the thread and node that attempted to process an authentication request.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Correlation ID – the unique ID associated with every request-response message pair.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Security Configuration – the associated security configuration.
- Method – the MBO operation used.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Outcome – the authentication outcome for the security check.
- Reason – the reason for authentication failure.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Error Log

Errors log data includes domain-level errors.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Correlation ID – the unique ID associated with every request-response message pair.

- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Connection Log

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

DOE Connection

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

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the subscription user for the package.
- Event Type – the DOE-C event type, such as Acknowledged, Duplicate Ignored, Exclude, No Response (from client or server), Packet Dropped, Registration Response, Resend (from client), Status Request (from client or server), DOE-C Subscription, and DOE-C Data Import.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Connection – the managed connection used. Its value is DOE for DOE-C logs.
- Client ID – the identifier for the DOE-C client.
- Physical ID – the DOE-C generated physical identifier registered with DOE at subscription.
- Subscription ID – the DOE-C generated subscription identifier registered with DOE at subscription.

- Logical Device ID – the DOE-C logical device identifier, generated by DOE and provided to DOE-C upon successful subscription.
- Message Direction – the DOE-C message direction, either client to SAP Mobile Server, or SAP Mobile Server to client.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.

Note: Payload and detail columns:

- Device ID – the core and administrative (MMS) device ID.
- Domain – the core and administrative (MMS) domain name.
- JSON Message Content – the messaging synchronization JSON message (payload). This is the SAP Mobile Platform-specific representation of the incoming DOE XML message in JSON format. DOE-C receives XML the payload from DOE in response, which is then parsed and converted to a JSON string and sent to the client.
- XML Message Content – the DOE SOAP messages (payload). This represents either an XML request in a format for sending to DOE by DOE-C, or an XML payload response received from DOE as applicable.
- Endpoint Name – the core and administrative (MMS) endpoint name.
- DOE server message ID – the SAP DOE reliable messaging server message ID.
- DOE client message ID – the SAP DOE reliable messaging client message ID.
- DOE-C server message ID – the DOE-C client-side SAP DOE reliable messaging server message ID.
- DOE-C client message ID – the DOE-C client-side SAP DOE reliable messaging client message ID.
- DOE-C method name – the DOE-C method being executed.
- DOE-C action name – the DOE SOAP action.
- Push to – the messaging asynchronous response queue.
- Address – the remote URL of the DOE server for this subscription (for example, `http://saphost:50015/sap/bc/DOE_ESDMA_SOAP?sap-client=600`).
- Log – the DOE-C subscription-specific log level.
- Extract Window – the DOE extract window for a subscription. This value determines the maximum number of unacknowledged "in-flight" messages allowed by the DOE reliable messaging protocol.
- PBI – the messaging synchronization "piggy backed import" setting for the subscription.
- Boolean property – indicates whether replay after-images can be piggy-backed onto `replayResult` and `replayFailed` messages (default is false).

JDBC Connection

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

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Connection – the managed connection used.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Payload and detail columns:

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

REST Connection

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

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- Connection – the managed connection used.
- URL – the URL associated with the managed connection.
- Action – the GET, POST, PUT, or DELETE action.
- Response Status – the response status code for the invocation.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Payload and detail columns:

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

SAP Connection

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

- Time – the time and date stamp for the log entry.

- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.
- MBO – the mobile business object used.
- Operation – the MBO operation.
- BAPI – the SAP BAPI used as the data source.
- Connection – the managed connection used.
- Properties – the list of name:value pairs.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Payload and detail columns:

- Parameters – input that was supplied to the operation; this will vary per request and operation (payload).
- Device ID – the core and administrative (MMS) device ID.
- Domain – the core and administrative (MMS) domain name.
- Endpoint Name – the core and administrative (MMS) endpoint name.
- SAP Host – the remote system hostname (if available).
- SAP User – the SAP user for the operation.

SOAP Connection

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

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Stage – the current stage of processing - START or FINISH.
- Package – the name of the package to which the subscription belongs.

- MBO – the mobile business object used.
- Operation – the MBO operation.
- Connection – the managed connection used.
- Service Address – the service address URL.
- Action – the SOAP action.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Payload and detail columns:

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

Push Log

Push logs include log data for all push notifications.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Source - the source of the log if its from the server or client.
- Correlation ID - the unique id associated with every request-response message pair.
- URN - not relevant.
- Log Level - not relevant.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Error – the error message if any.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.

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

Proxy Log

Proxy logs all data made to and from the Proxy server.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Source - the source of the log if its from the server or client.
- Correlation ID - the unique id associated with every request-response message pair.
- Request Type - the request type of the message.
- Request URL - the Gateway URL.
- HTTP Endpoint - the Gateway URL.
- Response Code – the response status code for the invocation.
- Log Level - not relevant.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.

Note: Additional detail columns:

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

Server Log

Server logs include logging data for SAP Mobile Server.

- Time – the time and date stamp for the log entry.

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

Dispatcher Log

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

Replication Log

Replication log data includes data for all replicated application data routed by the dispatcher.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Source - the source of the log if its from the server or client.
- Request URL - the Gateway URL.
- Response Code – the response status code for the invocation.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.

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

Messaging Log

Messaging log data includes data for all message-based application data routed by the dispatcher.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Source - the source of the log if its from the server or client.
- Request URL - the Gateway URL.
- Response Code – the response status code for the invocation.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Request Header Fields - the HTTP request header field contained in the application. For example, used by REST API-based application to create an application connection.
- Response Header Fields - the HTTP response header field communicated to the device from the server.

Service Log

Service log data includes application service data routed by the dispatcher.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.

- Source - the source of the log if its from the server or client.
- Response Code – the response status code for the invocation.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Request Header Fields - the HTTP request header field contained in the application. For example, used by REST API-based application to create an application connection.
- Response Header Fields - the HTTP response header field communicated to the device from the server.

Application Log

Application log data includes application specific messages, including Registration and Setting.

Registration Log

Registration log data includes registration-related application data.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Source - the source of the log if its from the server or client.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Auto Registraton - the automatic connection registration setting of the application, as determined by the `autoreghint` provisioning property.
- Security Configuration - the security configuration assigned to the application.
- Template - the application template used by the application.

Setting Log

Setting log data includes application settings information.

- Time – the time and date stamp for the log entry.
- Application ID – the unique identifier assigned to the registered application. Values may include a number, blank, or HWC, depending on the client type.
- Application Connection ID – the unique identifier for a user application connection.
- User – the name of the user associated with the application ID.
- Source - the source of the log if its from the server or client.
- Log Level – indicates the log level, if any, set on the client that controls what and how much should be logged.
- Thread ID – the identifier for the thread used to process the request.
- Node ID – the server node on which the request is received.
- Transaction ID – a unique ID that represents a transaction (one cycle of request-response) performed by the client or application.
- Root Context ID – a unique ID that represents a client/server session. A session can be thought of as a block that includes multiple requests from the client to the server.
- Operation – the MBO operation.
- Request Body - the application's HTTP request body field.
- Response Body - the server's HTTP response body field.

Agentry Application Logs

Agentry application logs record events and client-server messages for Agentry applications.

Events.log File

The `events.log` file records various Agentry Server events that occur at run time. Events include startup and shutdown, opening and closing connections to the application's system connections, loading of Client-Server communication components, memory usage, thread expansion, and used and available storage.

The events log persists until one of the following occurs:

- The log files are rolled by the Server, at which point they are moved to a backup folder named `Logs-Rolled`, in a sub-directory named for the date and time. By default, this occurs once every 24 hours provided the Agentry Server is running. With a change to the Server's configuration, this file may also be moved to a back up location based on file size.
- The Agentry Server is shut down and restarted. In this case, the events log file is moved to the backup folder `Logs-Rolled`, again in a sub-directory named for the date and time.

Unlike most other log files generated by the Agentry Server, the events log is always enabled. It cannot be disabled.

The following is an example of the contents of the `events.log` file:

```
11/28/2005 15:11:01, 1, 3, 7, Thr 1912, Server, Agentry Configuration,
   userKey, aaaa, File: agentry.ini,
   C:\Syclo\agentry\system.cpp#2103:DTLoggerHandler::badKey
11/28/2005 15:11:01, 0, 0, 2, Thr 1912, System Startup
11/28/2005 15:11:01, 1, 3, 8, Thr 1912, Invalid server license: Invalid User
   Key, C:\Syclo\agentry\server.cpp#444:System Startup
11/28/2005 15:11:01, 0, 11, 14, Thr 1912, TCP v4.0.0.0
11/28/2005 15:11:01, 0, 13, 14, Thr 1912, MID v4.0.0.0
11/28/2005 15:11:01, 0, 1, 4, Thr 1912, Agentry v4.0.0.0
```

Note: Each log message is written to a single line. Space restrictions in the example require some of the messages to wrap to multiple lines. Items in the example that are a part of the message above it are tabbed in. Each message begins with a timestamp.

The first column of each message contains the date and time of the event being logged. The next three columns, separated by commas, are the message type, message group, and the message ID. The message type is always a **0** or a **1**. A message type of **1** indicates an error message, while a message type of **0** indicates an informational message. The message type is followed by the thread ID, denoted by the text `Thr`, and finally the message itself.

Events Log Grid View

In the Grid View there are the following columns:

- Time: The timestamp of the message
- Message Type: THe message type, either informational (0) or error (1).
- Thread ID L The processing thread from which the message was generated.
- Message: The log message content.

Messages.log File

The `messages.log` file contains log items related to client-server messaging. Each message and response sent between the Agentry Server and Agentry Client is a log item in the `messages.log` file. The information in each item includes the date and time of the message, the Agentry Client's IP address, the type of message or response, and other message information.

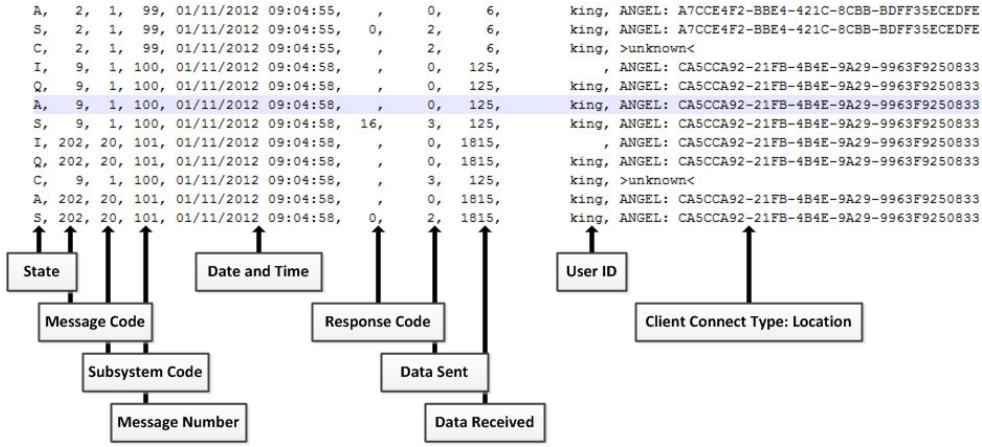
The messages log persists until one of the following occurs:

- The log files are rolled by the Agentry Server, at which point they are moved to a backup folder named `Logs-Rolled`, in a sub-directory named for the date and time. By default, this occurs once every 24 hours provided the Agentry Server is running. With a change to the Agentry Server's configuration, this file may also be moved to a back up location based on file size.
- The Agentry Server is shut down and restarted. In this case, the `messages.log` file is moved to the backup folder `Logs-Rolled`, in a sub-directory named for the date and time.

Unlike most other log files generated by the Agentry Server, the `messages.log` file is always generated by the Agentry Server and cannot be disabled.

Messages Log Format

The following is an excerpt from a typical `messages.log` file in the Text View, with each column of output labeled. These same columns are also presented in the Grid View. See the list of columns below the example. where there is a description of each output column.



- **Message State:** A single character indicating the state of the message in relation to the Agentry Server. See the *Message States* and *Push Message States* tables later in this section for more information.
- **Message Code:** A numeric value that identifies the type of message. See the table *Message Codes* later in this section for more information.
- **Subsystem Code:** Always either a 1 or 20. This is a deprecated value that will only vary for older SAP products built on Agentry 2.0 code libraries.
- **Message Number:** A number assigned by the Agentry Client that increments for each message sent.
- **Date and Time:** The date and time the message was received by the Agentry Server. This is always the date and time of the Agentry Server's host system.
- **Response Code:** A numeric code that indicates the response to the message. The message codes "S" and "R" each have a different set of possible response codes. For all other message codes, the response code column is blank.
- **Data Sent/Data Received:** Specifies a running total of the amount of data sent and received by the Agentry Server per message, with a value in bytes.
- **User ID:** Client ID of the user to which the message belongs. It is normal for the user ID to remain blank for messages with a message state of "I".
- **Client Location:** The IP address of the client device. It is normal for the Client Location to be unknown for messages with a message state of "C".

The following tables contain message state information for the `messages.log` file. There are two tables of message states. The first table is for synchronous transmissions. The second table is for asynchronous push messages only.

Table 46. Message States - Synchronous Communications

State	Name	Description
I	Incoming	Message in the process of being received from the client.
Q	Queued	Message is decoded, the user is identified, and the message is placed in one of the Agentry Server's work queues.
A	Active	A previously queued message is now being processed by the Agentry Server.
S	Sent Response	The Agentry Server sent information and/or an acknowledgement to the Client.
R	Received Response	A Client response was received by the Agentry Server.
C	Complete	The message processing is complete.

Table 47. Push Message States - Asynchronous Communications

State	Name	Description
O	Outgoing	A message is in the process of being sent to the Client.
T	Trying	The Agentry Server is attempting to connect to a Client
L	Linked	The Agentry Server has successfully connected to a Client.
W	Waiting	The Agentry Server failed in an attempt to connect to a Client. It will attempt the connection again.
R	Received Response	The Agentry Server has received a Client response.
S	Sent Response	The Agentry Server sent information and/or an acknowledgement to the Client.
C	Complete	The message processing is complete.
X	Cancelled	The Agentry Server cancelled the message to the Client.
F	Failed	The message has failed. The Agentry Server will not retry.

The following table lists the message codes, which indicate the type of message being processed or sent. These values are found in the Message Code column of the `messages.log` file.

Table 48. Log Message Codes

Code	Description
2	Client logout request
3	Client login request
7	Client user password change request
200	Transaction instance sent to the Agentry Server from the Client
201	Client has made a fetch processing request
202	Client system information message sent to Agentry Server
203	Client has requested an object be reloaded/refreshed
204	Client has requested the definition of an object be sent
205	Client has requested the definition of a fetch be sent
206	Client has requested the definition of a transaction be sent
207	Client has requested the definition of a screen set be sent
208	Client has requested the definition of an action be sent
209	Client has requested the definition of a rule be sent
210	Client has requested the definition of a report be sent
211	Agentry Server has pushed an object and/or messages to the Client
212	Client has sent an enable push message to the Agentry Server for the user
213	Client has requested the definition of a style be sent
622	Client has requested all complex tables be updated
623	Client has requested all data tables be updated

Rolling Agentry Message and Event Log Files

Each Agentry application has its own Agentry Server instance that runs on the SAP Mobile Server node. Each Agentry Server instance generates message and event logs for the application, which can be viewed through the Domains folder in SAP Control Center. Agentry message and event log files are automatically rolled based on the configuration of the Agentry application server instance. You can also manually roll the log files.

When you roll an Agentry application log file, the SAP Mobile Server renames the `messages.log` and `events.log` based on the current date and time, and copies them to the backup folders `mlog` and `elog`, respectively. The log files are located in *`SMP_HOME`*`\Servers\UnwiredServer\logs\<`*`agentryApplicationID`*`>`.

1. In the navigation pane of SAP Control Center, expand the **Applications** node and select the application.
2. In the administration pane, click the **Logs** tab.
3. Click **Roll log files**.
4. Click **Yes** to confirm.

Transport Archives

Exporting a package or application creates an archive file, which is used to transport development artifacts between servers.

MBO Package Export Archive

The archive file created when you export an MBO package contains deployment information, third-party libraries, and package properties and settings.

- Deployment information:
  - MBO model
  - Endpoint reference
- Third-party libraries:
  - Third-party JAR files
  - Classes the deployment unit uses
- Package properties and settings:
  - Domain name
  - Package name, type, status, and version
  - Security configuration
  - Subscription bulk-load timeout
  - Role mappings
  - Cache group settings
  - Synchronization group settings
  - Subscription template settings
  - Assigned application IDs
  - Server connection

See also

- *Transporting Artifacts Between Servers Using CTS* on page 221

Hybrid App Export Archive

The archive file created when you export a Hybrid App contains Hybrid App settings, context variables, matching rules, and application connection templates.

- The Hybrid App deployment file.
- The general settings of the Hybrid App, including display name, display icon, and description.
- The context variable key-value pairs for the Hybrid App.
- The matching rules for the Hybrid App.

  Note: All matching rule search expressions are exported as regular expression types. Other expresssion types such as `Begins with` or `Equals`. are exported as `Regular expression`.

- Information about the application connection templates that the Hybrid App is assigned to.

See also

- *Transporting Artifacts Between Servers Using CTS* on page 221

Application Export Archive

The archive file created when you export an application contains application metadata, and all application connection templates, customization resource bundles, MBO packages, Hybrid Apps, and native push notifications used by the application.

- Application metadata, such as archive type, archive version, application ID, application display name, application description, and the domains the application belongs to.
- All application connection templates related to the application. Each application connection template is stored as a separate XML file. The application connection template settings are stored as key-value pairs in the file.
- All customization resource bundle ZIP files used by the application.
- All MBO packages used by the application.
- All Hybrid Apps related to the application.
- All native push notification configurations used by the application.

See also

- *Transporting Artifacts Between Servers Using CTS* on page 221

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X