PUBLIC
SAP Work Manager
Document Version: 6.6 – 2024-09-24

SAP Work Manager Installation Guide

THE BEST RUN SAP

Content

Document History

Before you begin reading this guide, be sure that you have the latest version. Find the latest version at SAP Work Manager.

The following table provides an overview of the most important document changes.

Document	Date	Description of Changes
1.0	JUL 2021	Original release of the *SAP Work Manager Installation Guide*, version 6.6.

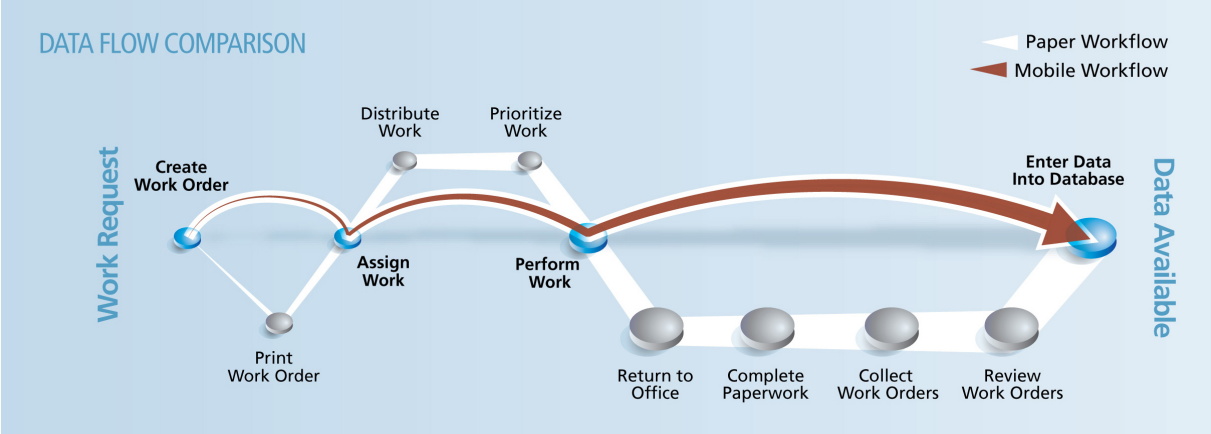
1 SAP Work Manager Overview

SAP Work Manager is designed to automate workflow and improve service with mobile work orders, notifications, and time management.

SAP Work Manager connects mobile employees with the data stored in an SAP system so they can better manage work and service requests.

Creating a Paperless Workflow

SAP Work Manager enables specific details such as asset, customer, or transaction histories and other critical information to be delivered to employees using mobile devices including smart phones, tablets, or laptop computers. Desktop systems are also supported. In turn, data is uploaded to an SAP system to generate follow-up work orders, status reports, customer invoices, charge backs and more.



SAP Work Manager Paperless Workflow

The paperless workflow process is as follows:

1. Workers log on to their mobile devices and download new work orders through a docking cradle or a real-time wireless connection.
2. Workers can then follow step-by-step job plans, verify work completed, and work status. Workers can also access additional details of a work order or notification to improve decision-making.
3. Workers update customer accounts, place orders, transfer or reassign work to another team member, or complete all required information and update to the SAP back-end system. The added data is accurate and usable, as SAP Work Manager enforces business rules on the mobile device. For example, by not allowing modifications or edits of objects if not allowed by the back end.
4. Workers transmit their updated or added data to the back end for any additional processing.

1.1 Additional Documentation on SAP Mobile Platform 3.0

Note that there are many features and functions supported by the SAP Mobile Platform. The guides and other technical information provided with the SAP Mobile Platform have fully documented all of the functional areas. You can access this information on the SAP Mobile Platform 3.0 help page.

> **Note**
>
> If you are upgrading to a new version of SAP Work Manager, also upgrade to the latest SAP Mobile Platform and SDK version. The SAP Mobile Platform team does not provide patches older than two releases.

Following are some of the key areas of functionality, presented with the sections of the SAP Mobile Platform documentation that describes it.

Security

Security features, including SSL authentication certificates for user authentication, client-side data encryption, and other security features are available for configuration. Some are set by default and can be adjusted. Other security features, however, you can enable and configure for your implementation environment. The following are sections where detailed information on security features is found within the SAP Mobile Platform documentation.

- Agentry Security
- Security Related Development Overview

Application Administration

Administration of all mobile applications deployed within the runtime environment, including SAP Work Manager, is accomplished using the SAP Cockpit. Administration includes cluster administration, security configuration, application administration, deployment, and several other tasks. The following lists some of the key sections on the SAP Cockpit. Read and become familiar with these topics, as and all of the topics available in this guide, as they may become pertinent later in your implementation.

- Guide: *Administration Overview*, sections of note:
  - Getting Started with Management Cockpit
  - Application Administration
  - Security Administration
  - Basic Deployment Model
  - Managing and Monitoring Applications

Development and Customization

Included with the SAP Work Manager application are the business logic and Java resources to modify the out of the box behavior of the application. These changes are made to the Agentry application project or by extending the Java logic provided for synchronization.

1.2 SAP Work Manager on the SAP Mobile Platform

SAP Work Manager is deployed on the SAP Mobile Platform as an Agentry application. The application is built with the Agentry Toolkit within the SAP EAM and service mobile app SDK.

> **Note**
>
> For detailed information on the Agentry Toolkit and the SAP EAM and service mobile app SDK, see Agentry App Development.

There are numerous components to the Agentry Toolkit, including the following:

- Runtime environment
- Agentry Editor
- Agentry client
- Agentry test environment (ATE)
- Development server (optional installation)

General overview information on each of these components is provided in this topic. For information the components, including usage, configuration, and other technical details, see the documentation provided with the SAP Mobile Platform and SDK.

Runtime Environment

The runtime environment is a production server system. Within this component, you can define one or more applications of different *archetypes*. An archetype refers to the different types of mobile applications, including the development paradigms, under which mobile applications are developed and deployed. The SAP Work Manager application is developed and deployed under the Agentry archetype. Therefore, it requires the definition of an Agentry application within the runtime environment into which it is deployed. You can modify the application by configuring options within the administration interface, the SAP Mobile Platform Cockpit.

Additionally, you can develop changes to out-of-the-box behavior of the application by using components of the Agentry toolkit within the SAP EAM and service mobile SDK.

SAP Mobile Platform Server

The server is provided within the SAP EAM and service mobile app SDK. It provides the same runtime functionality as the Agentry server component within the SAP mobile runtime environment. It is provided in the

Agentry toolkit to allow you to install the server without the need to install the entire runtime environment for development work. The development server is not intended for production use.

Agentry Plug-in to Eclipse

The Agentry plug-in to Eclipse provides a 4GL, point-and-click interface that allows developers to modify the SAP Work Manager application. The Agentry is provided in the SAP EAM and service mobile app SDK as a part of the Agentry toolkit. Agentry applications are stored within the Agentry in the Eclipse workspace as Agentry application projects.

Both the Agentry application project and the Java packages and projects are all managed within a single Eclipse workspace. The developer is presented with a single IDE in which the mobile application as a whole can be maintained and modified. See the document Agentry App Development in the SAP Mobile Platform documentation for details on working within this toolset.

SAP Work Manager Client

The SAP Work Manager client is provided for each of the client device types supported by the SAP Work Manager application. An installer for the SAP Work Manager client build is provided for Windows operating systems. For devices running iOS, the application is installed from the App store, or from your Apple Enterprise Server site. For Android devices, the `.apk` file is provided. You can install the `.apk` file from a Web server by navigating to its location within the local network of your environment.

The SAP Work Manager client is an executable run on the client device by the end user. The overall architecture of the Agentry toolkit allows for the development of a single application project, which can be deployed on multiple device types. The client processes the business logic developed in the Agentry and is deployed to the runtime environment. The client displays the user interface according to the native operating system of the device on which the SAP Work Manager is running.

The Agentry Test Environment

The Agentry Test Environment (ATE) is among the development tools provided in the Agentry Toolkit within theSAP EAM and service mobile app SDK. It is a highly useful tool for developers during the development cycle used for testing the client-side behavior and functionality of your mobile application. It includes numerous debugging and inspection tools to provide insight into the data, action execution, rule evaluation, and other aspects of the behavior of the client at runtime.

The ATE is not an emulator. However, it does posses the capability to mimic the behavior of all the supported client device types. Within the ATE is a full Agentry client. Part of the features of the ATE is the ability to select from a list of supported client platforms from within the Agentry archetype development paradigm. Once you select a platform, the Agentry client tells the SAP Work Manager server that it is a client of that type. The client then receives the user interface components for that platform.

1.3 What's Different with Agentry in the SAP Mobile Platform

The SAP Work Manager application is an Agentry application deployed within the runtime environment. If you were familiar with older releases of this application, which was deployed on the Agentry mobile platform, this topic provides information on the SAP Mobile Platform.

In addition to the information provided here, we recommend that you review the information available with the runtime environment, as well as the SAP EAM and service mobile app SDK. Many procedures employed in the configuration and deployment of the SAP Work Manager application are documented in those guides and manuals.

SAP Mobile Platform Server within the Runtime Environment

The server within the SAP Mobile Platform is now an application type within the runtime environment. If you are familiar with terminology from the Agentry Mobile Platform, you will recognize that the SAP Mobile Platform is the equivalent of the Agentry production server. From a functional standpoint there are no significant changes to the behavior of the server functionality. Data and business logic are served up to clients just as they were in the Agentry Mobile Platform. Data synchronization is handled in the same manner.

Differences involve how the settings for the server are configured. It is no longer possible to modify configuration files directly for applications deployed to the environment. Rather, you have two options for server configuration. The first option is to use the SAP Cockpit, which is the administration console for the server. Use the SAP Cockpit to modify any configurations that you previously would have made to the `Agentry.ini` file. All configuration settings are available within the SAP Cockpit.

The second option is to modify copies of the configuration files directly outside of the runtime environment. Then deploy the configuration files to the Agentry application within the environment. The deployment method is the required procedure for all configuration files in need of modification other than the `Agentry.ini` file. All of the other configuration files are not configurable within the SAP Cockpit.

Agentry Development Components in the SAP EAM and service mobile app SDK

The various development components of the former Agentry Mobile Platform are now provided as a part of the SAP EAM and service mobile app SDK, including:

- The development server (if installed)
- The Agentry plug-in
- The Agentry Test Environment
- The Agentry client installers

Each of these components is installed and used in the same manner as they were within the Agentry Mobile Platform.

SAP Cockpit Replaces Agentry Administration Client

Almost all functions of the Agentry administration client are now handled by the SAP Cockpit. In addition to configuration settings for runtime behaviors of the application itself, the SAP Cockpit also handles the following functional areas:

- Server logging and log file management
- Creating and managing clustered server environments
- Backup and restore of former server-side resources

Review the information provided with the SAP Mobile Platform, including guides and manuals, as well as any other publications, for information on these procedures and functional areas.

For more information and documentation on the SAP Mobile Platform, see the SAP Mobile Platform home page on the SAP Help Portal.

2 SAP Work Manager System Requirements Overview

The system requirements for the SAP Work Manager application are primarily driven by the requirements of the SAP Mobile Platform upon which it is deployed. Install the SAP Mobile Platform before installing the SAP Work Manager application. Meet all installation requirements of the SAP Mobile Platform.

Install the Mobile Add-On for ERP before installing the SAP Work Manager application.

> **Note**
>
> For full back-end system requirements, see the Mobile Add-On for ERP Installation Guide. If your system is installed below EHP7 SP14, see the Mobile Add-On 6.2 release page.

3 Deployment and Configuration of SAP Work Manager

The SAP Work Manager application is deployed to the SAP Mobile Platform as an executable file. The overall configuration and deployment process includes the following main tasks:

1. Run the setup program to configure the application settings.
2. Modify the configuration settings within certain configuration files.
3. Deploy the application into an Agentry application definition within the runtime environment.
4. Optionally, import the application into the Eclipse workspace containing the Agentry plug-in, resulting in the creation of an Agentry application project.

Before performing any of these tasks, first install the runtime environment and define an Agentry application within it.

> ⓘ Note
>
> User name and password security policies are defined in SAP ERP.

3.1 Installing the SAP Work Manager Application

Prerequisites

Address the following items before installing the application:

- The SAP Mobile Platform is installed and an Agentry application is defined within it. To define an Agentry application within this environment, see the documentation for the SAP Mobile Platform, specifically the topic Defining Applications.
- The Agentry is at least version sp13 pl06 (70.13.6).
- Depending on your set up, install either the Mobile Add-On for ERP or the Mobile Add-On for SAP S/4HANA to the SAP system.
- Install SAP Business Technology Platform Mobile Services. See the SAP Agentry Cloud Edition (ACE) Quick Start Guide for more information.

Context

The following procedure describes the steps to install the SAP Work Manager application to the runtime environment. The procedure also makes the SAP Work Manager application available for import into the Eclipse workspace as an Agentry application project.

You run the application executable file, SAP_WORK_MANAGER_66, during the procedure. The configuration tool provided with the application to set the implementation-specific configuration options is also run.

The following files are used in the implementation of this application. However, not all files are used for a given installation. Read the descriptions in the list to determine which is needed for your environment:

- `SAPWorkMgr66Deployment.exe`: The SAP Work Manager application for Windows deployments of the SAP Mobile Platform run time environment
- `SAPWorkMgr66MeterMgrDeployment.exe`: The SAP Work Manager application with the Meter Management component for Windows deployments of the run time environment
- `SAPWorkMgr66CustomerServiceDeployment.exe`: The SAP Work Manager application with the Customer Service component for Windows deployments of the run time environment

> **Note**
>
> If you are running SAP Mobile Platform on Linux, run the installation *steps 1-8* on Windows. Then use the SAP Cockpit to publish to the Linux-basedSAP Mobile Platform server.

Procedure

1. Double-click the appropriate `.exe` file for the runtime environment with the options related to the add-on component.
2. Launch the appropriate deployment executable for the SAP Work Manager application.

   The Welcome screen is displayed.
3. Click *Next* to continue the installation.

   The SAP Connectivity Information screen displays.
4. Enter the name of the SAP server. Enter the *Client* number and *System Number* the SAP Work Manager application uses to communicate with the SAP application server.
5. By default, the *S/4HANA Backend* box is not checked.
   - If unchecked, this means that the SAP system is not an SAP S/4HANA system and during the installation, a BAPI constants file is generated. The constants file contains an SAP table and field names, and is found in the `Java` folder.
   - If checked, it means that the SAP system is an SAP S/4HANA system. During installation, both a BAPI constants file and an SAP S/4HANA BAPI constants file are generated. Then, the SAP S/4HANA BAPI constants file overrides the BAPI constants file. The BAPI constants file contains an SAP table and field names while the SAP S/4HANA BAPI constants file contains SAP S/4HANA field names. These files can be found in the `Java` folder.

   > **Note**
   >
   > The Mobile Add-On for ERP (SP01) also supports SAP S/4HANA, on-premise Edition 1511 Feature Pack Stack (FPS) 01.
6. By default, the SAP Business Technology Platform box is not checked.
   - If unchecked, this means that the SAP system is not an SAP Business Technology Platform system and SAP Mobile Platform installation and configuration can proceed as normal.

- If checked, it means that the SAP system is an SAP Business Technology Platform system. ForSAP Mobile Platform installation and configuration instructions, see the SAP Agentry Cloud Edition (ACE) Quick Start Guide, *Step 7*.

Click *Next* to continue. The User Information screen displays.

7. Enter the *Service User Name* and *Service User Password*. The service user is an administrative user established as a proxy for all users.
8. If you are using push notifications, check the *Enable Push* checkbox and then enter the *Push User Name* and *Push User Password*. The push user is an administrative user established as a proxy for all users. In most cases, the push user can be the same user name and password as the service user. Click *Next* to continue.

> **Note**
>
> The Service User Password value entered in the previous steps is stored by the installer in the `JavaBE.ini` configuration file as plain a text value. Storing it as plain text makes it human readable. It is highly recommended that these values are secured using the Encrypt Password utility installed with the SAP Mobile Platform. You can run the Encrypt Password utility run after completing this procedure.

The Cloud Information screen displays.

9. If you are using push notifications with SAP BTP services, continue to the substeps. If you are not using push notifications with SAP BTP services, click *Next* to continue to the next step.

   The *Cloud Endpoint Hostname* field and the *Cloud Endpoint Application Name* field are used to configure the URL for the ERP backend to send push notifications to SAP Work Manager through the cloud instance.

   a. Specify the host for push notifications in the *Cloud Endpoint Hostname* field. The cloud endpoint hostname is the hostname of the cloud instance. For example, if the hostname is `https://instanceProvider-Tenant.domain/appname`, then the hostname is `instanceProvider-Tenant.domain`.

      If you do not type a value into the *Cloud Endpoint Hostname* field, the cloud instance attempts to determine the hostname value at runtime through its environment values and parameters.

   b. Specify the ID of the application in the *Cloud Endpoint Application Name* field. The ID of the application is defined in the Administration UI in your cloud instance.
   c. After filling in the fields, click *Next*.

   The Choose Install Location screen displays.

10. To specify the destination directory for the deployment ZIP file, either enter the path manually or click *Browse*. Select the location in the File Explorer dialog. Click *Install* to proceed with the installation. See the deployment ZIP file (example: `SAP WrkMgr50Deployment.zip`) in the directory specified in the last step. Click *Finish* to close the configuration application when the final screen is displayed.

> **Note**
>
> If you are running SAP Mobile Platform on Linux, run the installation *Steps 1 - 10* on Windows. Then use the SAP Cockpit to publish the Linux-based server.

11. Open the SAP Cockpit and publish the deployment ZIP file. For more information on the cockpit, see SAP Mobile Platform documentation.

    You have published the SAP Work Manager application. Your client devices can now access the application.

12. Install the SAP Mobile Platform certificate on the clients and perform an initial sync to the server using the appropriate server URL.

Results

You can now deploy the SAP Work Manager application to the runtime environment. It includes the application as delivered by SAP and the configuration settings in support of the communications between the mobile application and the SAP system. Use the contents of the folder where the archive was created as an import source to create an Agentry application project in Eclipse containing the Agentry plug-in.

Next Steps

Deploy the application to the runtime environment, specifically to the defined Agentry application within the runtime. For details on this procedure see the information provided with the SAP Mobile Platform, including the section Deploying Applications.

3.2 Configuring SAP Work Manager for the Time Zone of the SAP System

Context

This procedure describes how to configure the SAP Work Manager application to use the time zone in which the SAP back-end system resides. Date and time values from the client are converted from their local time zones to the time zone of the SAP system. The conversion is based on the settings by the SAP Work Manager application when the values are updated to the SAP system. Conversely, date and time values are converted from the time zone on the SAP system to the time zone on the client when being downloaded to the client by the SAP Work Manager application.

> ⚠ Caution
>
> Though not a common occurrence, operating system manufacturers sometimes change the name used by the operating system for a given time zone through a service pack or patch update. If a time zone name is changed due to a service pack or patch update, check and ensure that none of the configured time zone names or time zone aliases are affected.

> → Tip
>
> Daylight Savings Time (DST) is respected in these conversions. Certain time zones do not respect DST and others may have different start and end dates concerning the observance of DST. Therefore, converting from one time zone to another where differences in DST observance exist can result in unexpected, but otherwise correct, values.

Procedure

1. Open the application in the Cockpit and ensure that you are on the *Back End* tab.
2. In the *Time Zone Name* field, set the following attribute to the time zone that the SAP system is set to: `timeZoneName=SAP's Time Zone Name`

   Example: SAP Server Time Zone is: W. Europe Standard Time

   Set `timeZoneName` **As:**

   `timeZoneName=W. Europe Standard Time`

   The time zone name value must match the time zone value on the operating system of the SAP server exactly. The time zone setting can be found in the registry setting of the SAP server:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones
   ```

   The "Current time zone" equivalent time zone name can also be used. For example, "Eastern Standard Time" can be used instead of Eastern Time Zone.
3. Click the *App Specific Settings* tab to fill out the *Time Zone Alias* field. Add one entry for each potential time zone where the client devices may be used in the following format:

   `client time zone name=server time zone name`. For example:

   `Central European Summer Time=W.Europe Standard Time`

   Where `Central European Summer Time` is the name the client device reports as its time zone and `W. Europe Standard Time` is the name the operating system that the runtime of the host system uses for the same time zone. You can add multiple entries to this section as long as the client devices are used in multiple time zones within the same implementation.

   The time zone aliases are used to reconcile the time zone name that the client reports it is in. The time zone name that the client reports may be different from the name used by the runtime environment. It is from the time zone of the client to the time of the SAP system to which the date and time values are converted. For this reason, the SAP Work Manager application must know how a time zone is identified by the client device. Therefore, there is a need for time zone aliases. The need for time zone aliases is especially true for client devices running a different operating system from the host system of the runtime client. However, it can also be necessary even when both are in the same operating system family (example: Windows mobile client device and Windows server).

   After a sync, the time zone name of the client can be seen from the user log. Note that Agentry is now using Olsen strings for time zones on Android and iOS clients. The advantage in using Olsen strings is that they are always in English. Olsen strings cut out the need for other language entries that would be required for customers who have multilanguage clients.

   The example shown demonstrates the following scenario: French WPF clients in Eastern Standard Time, multilanguage iOS clients in Central Standard Time, and several WinCE clients in Europe Standard Time and CST.

   {·} Sample Code

   ```
   [TimeZoneAlias]
   Central European Standard Time = W. Europe Standard Time
   Central European Summer Time = W. Europe Standard Time
   Central Daylight Savings Time = Central Standard Time
   Central Standard Time = Central Standard Time
   ```

```
Heure Avancee de l'Est = Eastern Standard Time
America/Chicago = Central Standard Time
```

4. Click *Save* to save the SAP Work Manager application in the Cockpit.

Results

The time zone alias(es) necessary to perform the proper time conversions are now configured.

3.3 Adding the Ability to Use Special Characters in Passwords

In SAP Work Manager, if a user attempts to change their password to one with a special character, such as €, or to log in with a password containing a special character, the password is rejected as invalid. Passwords containing special characters are allowed and are verified as working in the SAP GUI.

You can configure the usage of special characters in passwords in SAP Work Manager. Ensure that the following prerequisites are met:

- Access to the SAP Mobile Platform Admin console that is hosting the SAP Work Manager application
- Access and ability to edit the `JavaBE.ini` file
- Understanding of the back end SAP code page configuration

The JCo connection library used by SAP Work Manager defaults to SAP codepage1100, as all ABAP systems support this code page. However, not all symbols are supported on code page 1100.

If your system is configured for Unicode, or for a different code page, specify that information in the `JavaBE.ini` file. Determine from your back end SAP administrator how your system is configured, then follow the appropriate instructions for whether you are using a Unicode or non-Unicode configuration:

Unicode Systems

Browse to the server directory containing the `JavaBE.ini` file and open the file for editing. Add the following to section labeled `[JCO3_CUSTOM_PROPERTIES]`:

```
jco.client.pcs=2
```

Save the file and restart the SAP Work Manager application. You can now log in using a password with special characters.

Non-Unicode Systems

Browse to the server directory containing the `JavaBE.ini` file and open the file for editing. Add the following to section labeled `[JCO3_CUSTOM_PROPERTIES]`:

```
jco.client.codepage=<SAPcodepage>
```

Save the file and restart the SAP Work Manager application. You can now log in using a password with special characters.

3.4 Connecting SAP Work Manager to a Distributed SAP Environment

In order for the application to connect to a database that is deployed in a distributed environment, change a property in the `LOGON_METHOD` in the `JavaBE.ini` file.

Procedure

1. Open the `JavaBE.ini` configuration file in a standard text editor. The `JavaBE.ini` file modified in this procedure is located in the directory where the SAP Work Manager application was extracted.
2. Locate the section `[LOGON_METHOD]`:

   ```
   [LOGON_METHOD]
   ;  USER_AUTH if standard UID or Password authentication is used
   ;  USER_AUTH_GLOBAL if pooled connections using single UID or Password is used
   ;  USER_AUTH_GROUP if UID or Password authentication with SAP Message Server
   ;   (load balancing) is used
   ;  USER_AUTH_SSO if SSO2 ticket authentication with SAP Portal Server is used
   ;  USER_AUTH_CUSTOM for a custom logon module setup
   LOGON_METHOD=USER_AUTH
   ```

3. Set the `LOGON_METHOD` property to `USER_AUTH`.
4. Save and close the `JavaBE.ini` file.

3.5 Connecting SAP Work Manager to a Load Balanced SAP System

Context

If the SAP Work Manager is going to connect to a database that is deployed in an environment with load balancing, follow the procedure in this section to configure the server for a distributed environment.

The `JavaBE.ini` file modified in this procedure is located in the directory where the SAP Work Manager executable resides.

Procedure

1. Open the `JavaBe.ini` configuration file in a standard text editor.
2. Locate the section `[LOGON_METHOD]`:

```
[LOGON_METHOD]
; USER_AUTH if standard UID or Password authentication is used
; USER_AUTH_GLOBAL if pooled connections using single UID or Password is used
; USER_AUTH_GROUP if UID or Password authentication with SAP Message Server
;   (load balancing) is used
LOGON_METHOD=USER_AUTH
```

3. Set the `LOGON_METHOD` property to `USER_AUTH_GROUP`.
4. Locate the section `[GROUP_LOGON]`:

```
[GROUP_LOGON]
; referenced when LOGON_METHOD=USER_AUTH_GROUP
; individual user authentication using an SAP Message server, which
distributes
; client connections among a "group" of SAP application servers based on load
; balancing criteria
;
; host name or IP address of SAP Message Server
MESSAGE_SERVER=
GROUP_NAME=
SYSTEM_ID=
CLIENT=
```

5. Add `SHAREDCONNECTIONS=` to the list of parameters in `[GROUP_LOGON]`. The list is as follows:

```
MESSAGE_SERVER=
GROUP_NAME=
SHAREDCONNECTIONS=
SYSTEM_ID=
CLIENT=
```

6. Set the properties for the `[GROUP_LOGON]` to the following parameters:
   a. Set the `MESSAGE_SERVER` to host name or IP address of the SAP message server.
   b. Set the `GROUP_NAME` to the name of the group of application servers.
   c. Set the `SYSTEM_ID` to the name of the SAP system or the SAP ERP name.
   d. Set the `CLIENT` to the client number, which the SAP Work Manager application uses to communicate with SAP ERP.
7. Save and close the `JavaBe.ini` file.

3.6 Connecting SAP Work Manager to the Agentry Component

Prerequisites

Ensure the following are met before performing the procedure:

- Ensure that you have a provisioned Agentry account. For more information, see Provisioning an Agentry Account.
- Set up your customer account. For more information, see Set Up Customer Accounts for Agentry.
- Define your security settings. For more information, see Defining Security Settings for Agentry Applications.

Context

> **Note**
>
> Some settings may not be relevant to your application. For example, if your application does not use push, push settings are not available to you.

If the SAP Work Manager application was installed using the SAP Business Technology Platform Mobile Services option, you can configure the Agentry component through the `JavaBE.ini` file.

The `JavaBE.ini` file is located in the directory where the SAP Work Manager executable resides.

Procedure

1. Open the `JavaBE.ini` configuration file in a standard text editor.
2. Locate the `[JCO]` section:

```
[JCO]
    CLASS=JCO3
    CONNECTION_TYPE=CLOUD
```

3. Ensure the `LOGON_METHOD` in the `[LOGON_METHOD]` section is set to `USER_AUTH`. Make any changes to the type of logon in the configured destinations.
4. Make any configuration changes necessary. Save and close the `JavaBE.ini` file.
5. **Add a virtual hostname and internal hostname for your ERP back end using the SAP Cloud Connector Administration tool. Note these virtual hostnames, as you need them for future steps in this procedure.**
   a. Click the *Add* icon.
   b. Select *ABAP System* for *Backend Type*.
   c. Select *RFC* for *Protocol*. If you are using principal propagation, select *RFC_SNC*.

d. Depending on your system setup, select either *With load balancing* or *Without load balancing*.
   - If you chose *With load balancing*, add the hostname of the *Message Server* and the *System ID*.
   - If you chose *Without load balancing*, add the hostname of the *Application Server* and the *Instance Number*.

e. Add the corresponding virtual host values for the *Application Server* and the *Instance Number*. The virtual host values can be the same as the hostname values.

f. Optionally, add a *Description* and click *Finish* to add the virtual hostname and internal hostname.

6. Select your newly created hostmap, and in the *Accessible Resources* table, click the *Add* icon and add the following prefixes:
   - `/SMERP/`
   - `/SMFND/`
   - `/SYCLO/`
   - `/SMISU/`

   > **Note**
   >
   > Only add `/SMISU/` if the Meter Management component is installed.

   Click *Save* to save the prefixes to the hostmap.

7. Open the SAP Business Technology Platform Cockpit for the tenant account, if it is not already open. Select the *Connectivity* > *Destinations* tab on the left side of the window.

8. Add three new destinations with a *Type* of *RFC*, as seen in the following three examples:

   > **Note**
   >
   > The following examples are primarily for systems without load balancing. See the bolded lines for additions or deletions for load balanced systems.

   - **DESTINATION_SERVICE:** Used for the service user and repository requests
     - Type: RFC
     - User: ServiceUser
     - Password: serviceUserPassword
     - jco.client.client: 800
     - jco.client.lang: EN
     - jco.client.ashost: virtual.server.hostname.from.the.cloud.connector **[Remove for load-balanced systems]**
     - jco.client.sysnr: 23 **[Remove for load-balanced systems]**
     - jco.destination.pool_capacity: 5
     - jco.destination.peak_limit: 10
     - **jco.client.mshost: virtual.message.server.hostname.from.the.cloud.connector [Add for load-balanced systems]**
     - **jco.client.r3name: ABC [Add for load-balanced systems]**
     - **jco.client.group: PUBLIC [Add for load-balanced systems]**
   - **DESTINATION_PUSH:** Used for push users
     - Type: RFC
     - User: PushUser

- Password: PushUserPassword
- jco.client.client: 800
- jco.client.lang: EN
- jco.client.ashost: virtual.server.hostname.from.the.cloud.connector **[Remove for load-balanced systems]**
- jco.client.sysnr: 23 **[Remove for load-balanced systems]**
- jco.destination.pool_capacity: 10
- jco.destination.peak_limit: 100
- **jco.client.mshost: virtual.message.server.hostname.from.the.cloud.connector [Add for load-balanced systems]**
- **jco.client.r3name: ABC [Add for load-balanced systems]**
- **jco.client.group: PUBLIC [Add for load-balanced systems]**

> **Note**
>
> Select only one of the following `DESTINATION_SESSION` choices listed, depending on whether your system is configured for standard authentication or principal propagation.

- **DESTINATION_SESSION:** Used for user logins, standard authentication
  - Type: RFC
  - User: ServiceUser
  - Password: serviceUserPassword
  - jco.client.client: 800
  - jco.client.lang: EN
  - jco.client.ashost: virtual.server.hostname.from.the.cloud.connector **[Remove for load-balanced systems]**
  - jco.client.sysnr: 23 **[Remove for load-balanced systems]**
  - **jco.client.mshost: virtual.message.server.hostname.from.the.cloud.connector [Add for load-balanced systems]**
  - **jco.client.r3name: ABC [Add for load-balanced systems]**
  - **jco.client.group: PUBLIC [Add for load-balanced systems]**
- **DESTINATION_SESSION:** Used for user logins, principal propagation
  - Type: RFC
  - User: [blank]
  - Password: [blank]
  - jco.client.client: 800
  - jco.client.lang: EN
  - jco.client.ashost: snc.secured.virtual.hostname.from.cloud.connector
  - jco.client.sysnr: 23
  - jco.destination.auth_type: PrincipalPropagation

3.6.1 Connecting SAP Work Manager Agentry Cloud Edition to the Back End using SAML Identity Provider

Mobile applications developed using Agentry, such as SAP Work Manager, use SAML 2.0 for identity authentication.

Prerequisites

- Agentry is deployed on SAP Business Technology Platform Mobile Services
- SAP BTP services Agentry Cloud Edition is 2016 or above. Note that SAML authentication is not supported on on-premise systems when using SAP Mobile Platform.
- Agentry SDK is 7.3 or above.
- SAML authentication requires SAP Secure Network Communication (SNC)
- The RFC SNC protocol must be used when configuring Agentry Cloud Edition to on-premise mapping in the Cloud Connector

Before starting this procedure, create and deploy the mobile application in SAP Business Technology Platform Mobile Services using the default authentication method. See the Connecting SAP Work Manager to the Agentry Component [page 20] for detailed instructions. Once the app is deployed, ensure that you can successfully connect from the Agentry client using your SAP user ID and password.

Context

Procedure

1. **Configuration changes to the Agentry application:**
   a. Open the `JavaBE.ini` configuration file in a standard text editor.
   b. Locate the `[JCO]` section and ensure the configuration is as follows:

```
[JCO]
    CLASS=JCO3
    CONNECTION_TYPE=PRINCIPAL_PROPAGATION
```

   c. Ensure the `LOGON_METHOD` in the `[LOGON_METHOD]` section is set to `USER_AUTH`.
   d. Locate the `[JCO3]` section and ensure the configuration is as follows:

```
[JCO3]
    ;Please map the destination names configured in BTP Cockpit
    DESTINATION_SERVICE_NAME=<DESTINATION_SERVICE>
    DESTINATION_PUSH_NAME=<DESTINATION_PUSH>
    DESTINATION_SESSION_NAME=<DESTINATION_SESSION>
```

e. Locate the `[USER_AUTH]` section and ensure the configuration is as follows:

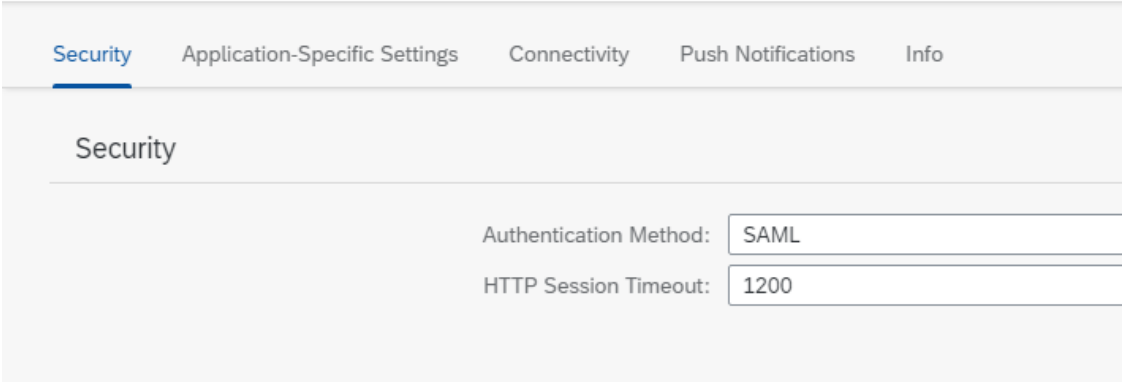
```
[USER_AUTH]
    ;USER_AUTH section configuration for SNC/SAML
    BYPASS_USERID_CHECK=true
    ALLOW_USERNAME_REMAPPING=true
```

f. *Save* your changes. Create the export and publish the definitions to the mobile app in SAP BTP services.

> **Note**
>
> Refer to the main procedure, Connecting SAP Work Manager to the Agentry Component [page 20], for additional information, if required.

2. **Security configuration so the application can use SAML:**
   a. Log on to the SAP BTP cockpit. Navigate to the subaccount for the Neo environment.
   b. Navigate to *Services* > *Mobile Services* > *Go to Service*.

      SAP BTP services is launched.
   c. Navigate to the mobile application you created by going to *Mobile Applications* > *Agentry* > *Your Mobile Application*.
   d. Navigate to *Assigned Features* > *Agentry Applications* and configure the *Security* as follows:
      - Authentication Method: SAML
      - HTTP Session Timeout: 1200 (default is 20 minutes, which is 1200 seconds)



   e. Configure the destinations you added previously using the Connecting SAP Work Manager to the Agentry Component [page 20] procedure. The following example uses *DESTINATION_SESSION_NAME*. Configure *DESTINATION_SERVICE_NAME* and *DESTINATION_PUSH_NAME* in the same way.
      - **Example configuration:**
        - Name: <DESTINATION_SESSION_NAME>
        - Type: RFC
        - Description: <description>
      - **Additional parameters:**
        - jco.client.client: <SAP client>
        - jco.client.lang: EN
        - jco.destination.auth_type: PrincipalPropagation

- jco.destination.peak_limit: 100
- jco.destination.pool_capacity: 10

For load balanced systems, use Message Server:

- jco.client.mshost: <message server host from Cloud Connector>
- jco.client.msserv: <message server>
- jco.client.group: <group name>
- jco.client.r3name: <SID>

For non-load balanced systems, use Application Server:

- jco.client.ashost: <server host name from Cloud Connector>
- jco.client.sysnr: <system number>

3. **Configure the cloud to on-premise mapping using RFC SNC:**

> ### Note
>
> - Choose *Protocol: RFC SNC* to connect to the back end.
> - The back end must be properly configured to support SNC connections.
> - The SNC configuration must be provided in the Cloud Connector.

Once you've configured the cloud to on-premise action control, you must configure the function modules. Configure the following *prefixes* as part of the *Resources* to the cloud to on-premise host map:

- /SMERP/
- /SMFND/
- /SYCLO/
- /SMISU/
- Z (allows custom RFC specific to your implementation)

Note that if you use a service user to start and run the Cloud Connector, that user account must be granted admin privileges.

For additional information, see the Configuring Principal Propagation topic in the *SAP BTP Connectivity for the Neo Environment* guide.

4. **Configure the SAP back end:**

> ### Note
>
> A system restart may be required for RZ10 and RZ11 parameter changes to take effect.

For additional information, see the Configure Principal Propagation for RFC topic in the *SAP BTP Connectivity for the Neo Environment* guide.

a. **STRUST configuration:** Use transaction *STRUST* to import the certificates from the Cloud Connector

> ### Note
>
> For more information, see the STRUST Configuration topic in the *Administration Guide to Implementation of SAP S/4HANA with SAP Best Practices* guide.

b. **Configure rule-based certificate mapping:** Use transaction *CERTRULE* to import the Cloud Connector principal progagation certificate and to configure the assertion rule and exceptions.

> **Note**
>
> For more information, see the Rule-Based Mapping of Certificates topic in the *SAP BTP Connectivity for the Neo Environment* guide.

3.7 Testing Connectivity to SAP

Prerequisites

Before performing this connection test, address the following items:

- Install the SAP Work Manager server.
- Ensure that the configuration files `Agentry.ini` and `JavaBE.ini` contain the settings as described in the server installation procedure.
- The person performing this procedure should log into the host system with administrative privileges.
- This procedure is only applicable to Windows installations of the SAP Work Manager server.

Context

Use the following procedure to confirm the SAP Work Manager server is able to communicate with the SAP ERP system. The following procedure is optional and you do not have to perform it as part of normal production duties. Rather, the procedure is provided to diagnose connectivity issues between the SAP Work Manager server and the SAP ERP system.

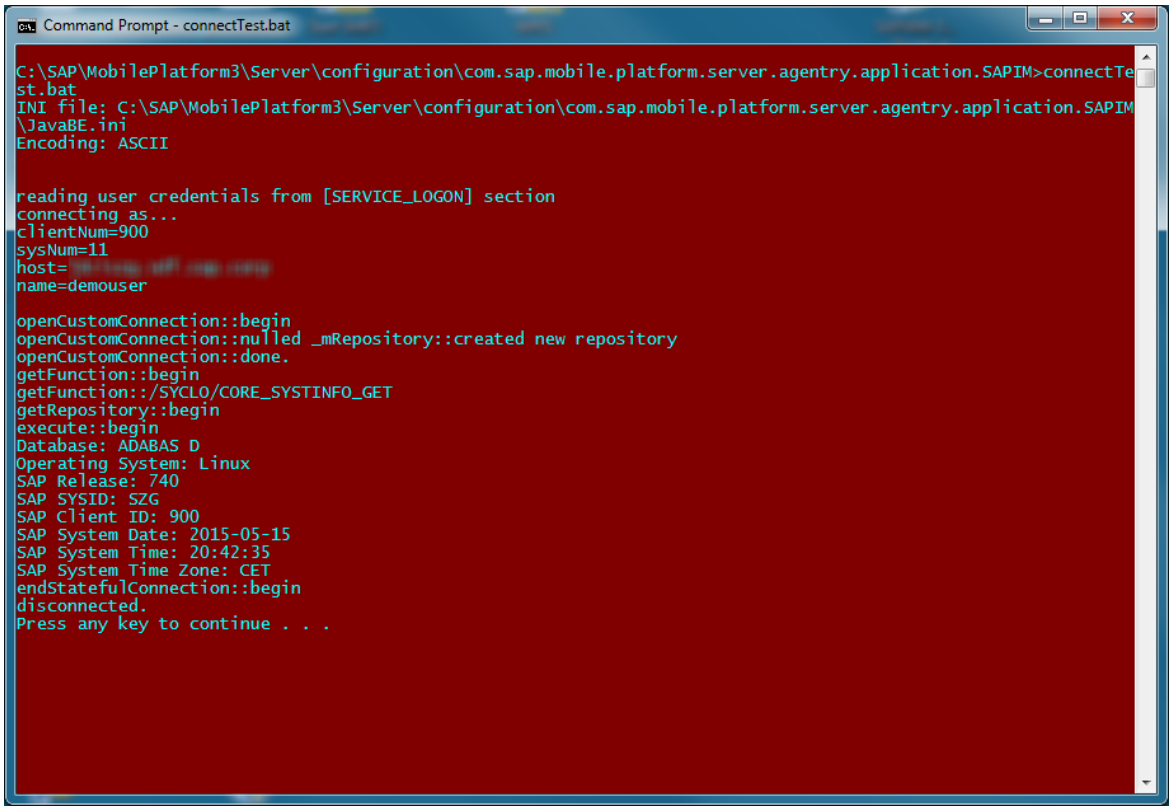
To confirm that the SAP Work Manager server can communicate with the SAP application, take the following steps:

Procedure

1. Open a command prompt window and change to the server directory of the SAP Work Manager application. Run the following script to test if the server can communicate with the SAP ERP system:

   ```
   connectTest.bat
   ```

If you configured the communications between the SAP Work Manager server and the SAP ERP system correctly, the script outputs the following:



Verify that the settings of the SAP Work Manager server are all correct before continuing.

2. Click the server icon or follow the start service instructions to start the SAP Work Manager server. Verify that no error messages are displayed, including pop up messages or log messages displayed on the user interface.

   - If no errors are encountered, the SAP Work Manager server was able to open communications with the SAP ERP system.
   - If you encountered any errors, continue on to *Step 3* of this procedure.

3. Go to the Service Marketplace Support Packages and Patches and find the SAP Mobile Platform SDK for your software release. Download the ZIP file for your release.
4. In the ZIP file, navigate to the *Modules* > *AgentryEditors* folder. Extract the `Agentry-v5.jar` file and place it in the *Java* folder of the SAP Work Manager server.
5. Open the `connectTest.bat` file, located in the SAP Work Manager server directory in a text editor. Make sure the class path reflects the location of the `sapjco3.jar` file and the `Agentry-v5.jar` file.

   A class path is shown here as an example:

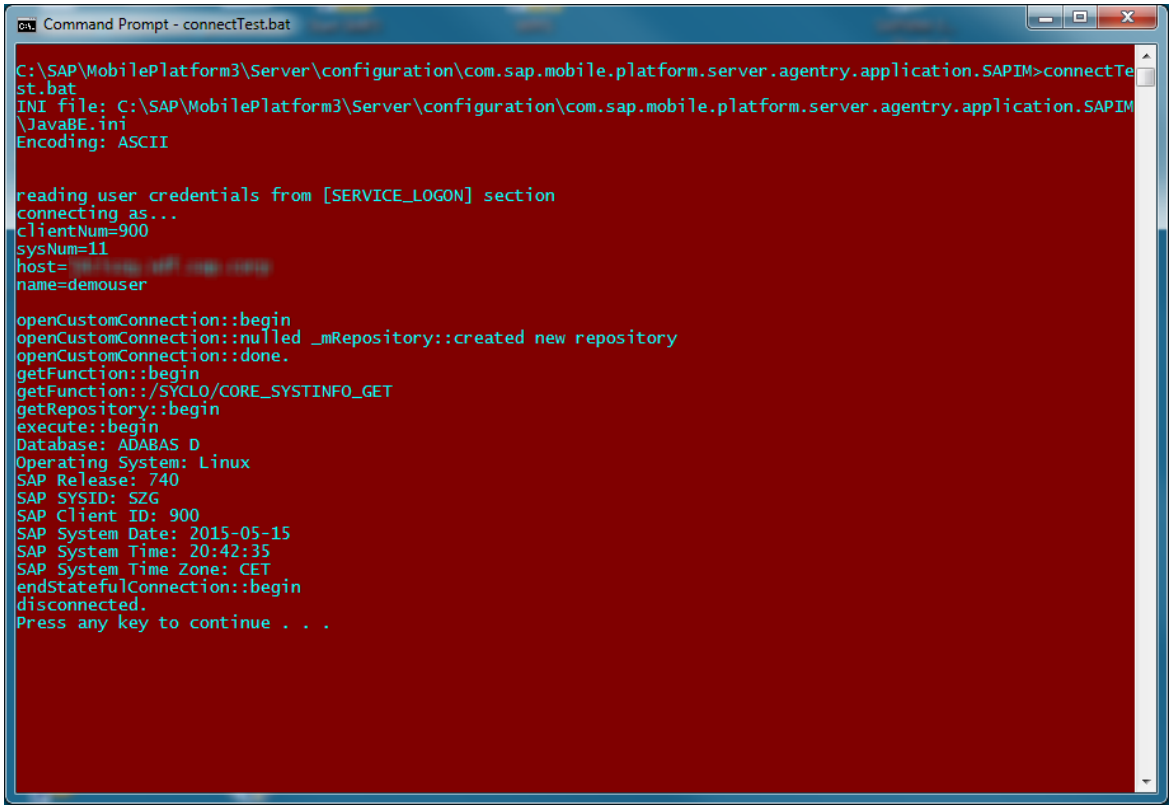
   Sample Code

   ```
   set classpath=./Java/Agentry-v5.jar;./Java;./Java/<mobile-
   applicationx.x.x>.jar;./
   Java/common-20150501.jar;./sapsso.jar;./sapjco3.jar;./ini4j.jar;;
   ```

6. Open a command prompt window and change to the directory of the SAP Work Manager server. Run the following script to test if the server can communicate with the SAP ERP system:

```
connectTest.bat
```

If you configured the communications between the SAP Work Manager server and the SAP ERP system correctly, the script outputs the following:



Verify that the settings of the server are all correct before continuing.

7. Click the server icon or follow the start service instructions to start the SAP Work Manager server. Verify that no error messages are displayed, including pop up messages or log messages displayed on the user interface of the server.

   - If no errors are encountered, the SAP Mobile Platform was able to open communications with the SAP ERP system.
   - If you encountered any errors, review this procedure and verify that you followed each step correctly. Pay particular attention to the configuration settings you modified in the configuration files for the SAP Mobile Platform.

3.8 Setting up an SAP Mobile Platform Development Environment

Starting with SAP Mobile Platform 3.0, the Agentry product installers create the production ZIP file. The ZIP file includes all configuration files to upload to the SAP Management Cockpit. Now, the configuration of the Agentry development server is the responsibility of the developer.

Context

Previously, in Agentry 6.x releases and prior, installation was a standalone server and the installer created both development and production servers if desired.

Follow these steps to set up a development server on the SAP Mobile Platform.

Procedure

1. Ensure that the Agentry application is defined within the SAP Mobile Platform. To define an Agentry application within this environment, see the documentation for the SAP Mobile Platform, specifically the topic Defining Applications.
2. Run the product installer to create the production ZIP file. The name of this file varies depending on the product that you are installing.
3. Unzip the production ZIP file to a temporary location.
4. Copy all the files from the unzipped folder, minus the application folder, to your application directory, under the SAP Mobile Platform server configuration folder (`\MobilePlatform3\Server\configuration\[YourApplicationDirectory]`)

   > **Note**
   >
   > This folder was created by the Management Cockpit when you created your application. The application directory name matches the name you give your application.

5. Edit the `Agentry.ini` file to change the key:

   ```
   developmentServer=True
   ```

6. Import the definitions into the Agentry Editor from the directory where you unzipped the production ZIP file.
7. Publish as development to the application directory under the server configuration folder.

   Your development server is populated with the application definitions.
8. Restart your Agentry application, or the entire server.
9. Verify your application in the Cockpit and make any additional changes, such as *Time Zone*, *urlPath*, or other configuration options. If any changes are made, restart the Agentry application or the server.
10. Verify that the Agentry application is running by viewing the standard `I am here!` text. Type in the URL of the application into a browser.

A Web page with the text displaying `I am here!` at the top appears if the application is running. If the application is not running, an error message appears.

11. Connect your ATE or client to the server.

3.9 Additional Parameter Settings

Additional parameters and globals for the SAP Work Manager application can be set and configured using the ConfigPanel.

For information on using the ConfigPanel as well as all the parameters and globals contained within, see the SAP Work Manager Configuration Guide.

4 Importing the SAP Work Manager Project into the Eclipse Workspace

Prerequisites

Address the following items before performing this procedure:

- Determine the source of the application project you are importing in this procedure and that you have access to that source.
- Verify that the application project source was created with the same or earlier version of the SAP Mobile Platform. You cannot import projects, export files, or published applications created with a later version into an earlier version.
- Ensure that Eclipse is installed. For information on installing Eclipse, see the following topics:
  - Agentry App Development
  - Installing the Eclipse IDE and Agentry Editor Plug-In
- Verify the workspace in which you are importing the project is the currently opened workspace in Eclipse.
- Determine a project name, or how the project is listed in the Eclipse workspace. The project name is required information entered in the import process.
- Determine a value for the *Name* attribute of the application definition. The value is information required during the import process.
- Though not required, it is strongly recommended that you install the SAP Mobile Platform. A development server is necessary when defining any of the synchronization logic within the project after it is created.

Context

This procedure describes the steps involved in importing an application project into the current Eclipse workspace. When this procedure is complete, a new Agentry application project is created in the current Eclipse workspace. This project contains the definitions and application components found in the import source.

Note that this process excludes any related projects for the source application that may reside in the workspace of the source project, such as Java development projects and related packages. Import these related projects and components according to the process that matches that project type, using tools found in Eclipse. Whether the Agentry application project is imported before or after other related projects is unimportant. However, import or configure all items before modifications are made.

This procedure accomplishes the following:

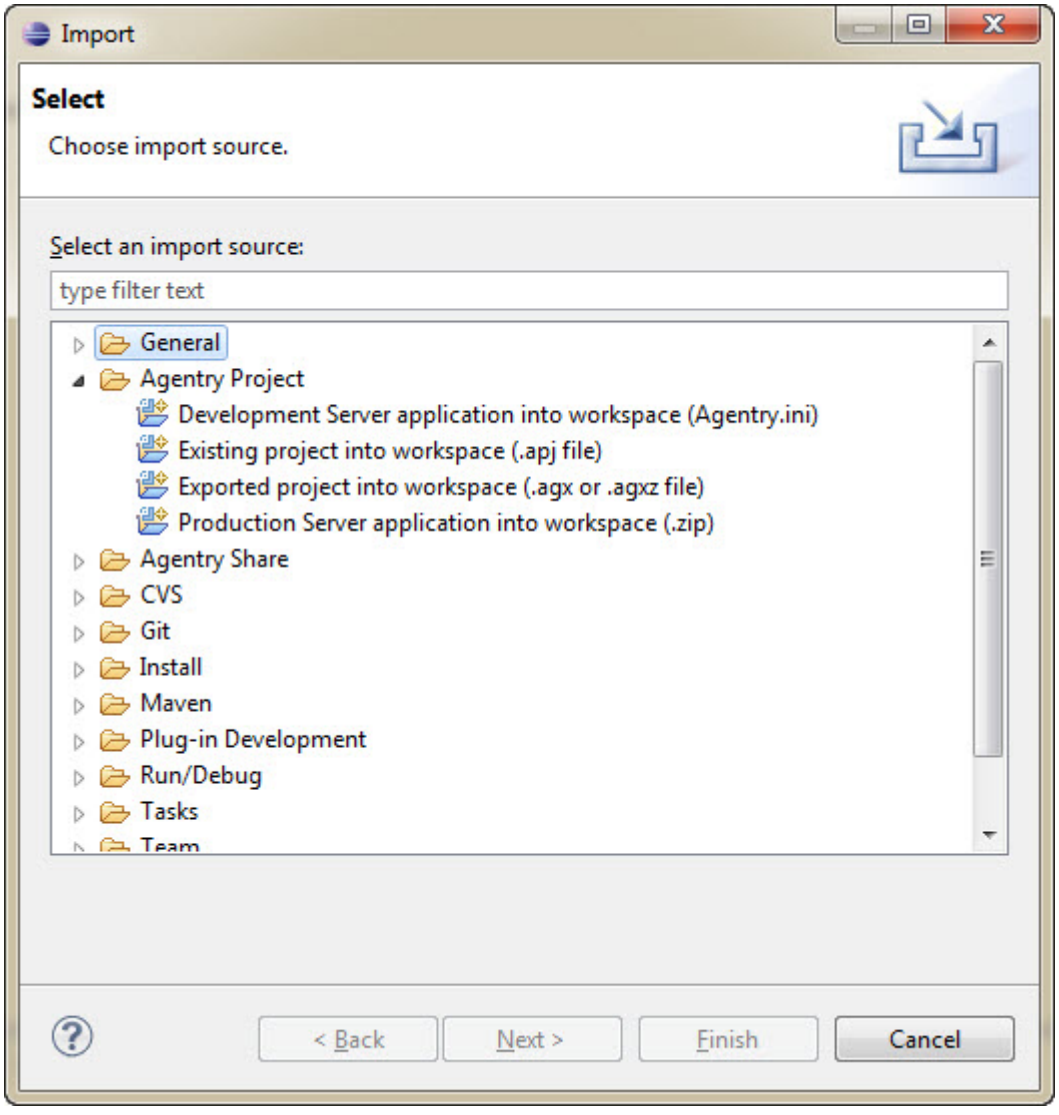
- Checks out an Agentry application project from an Agentry share repository and creates a new local project based on the top revision within that repository.
- Migrates an application project from one workspace to another.
- Upgrades an application project or application export file created in a previous release of the SAP Mobile Platform.

- Restores or recovers an application project from an archived project or export file.
- Creates, restores, or recovers an application project from a mobile application published to the SAP Mobile Platform.

Procedure

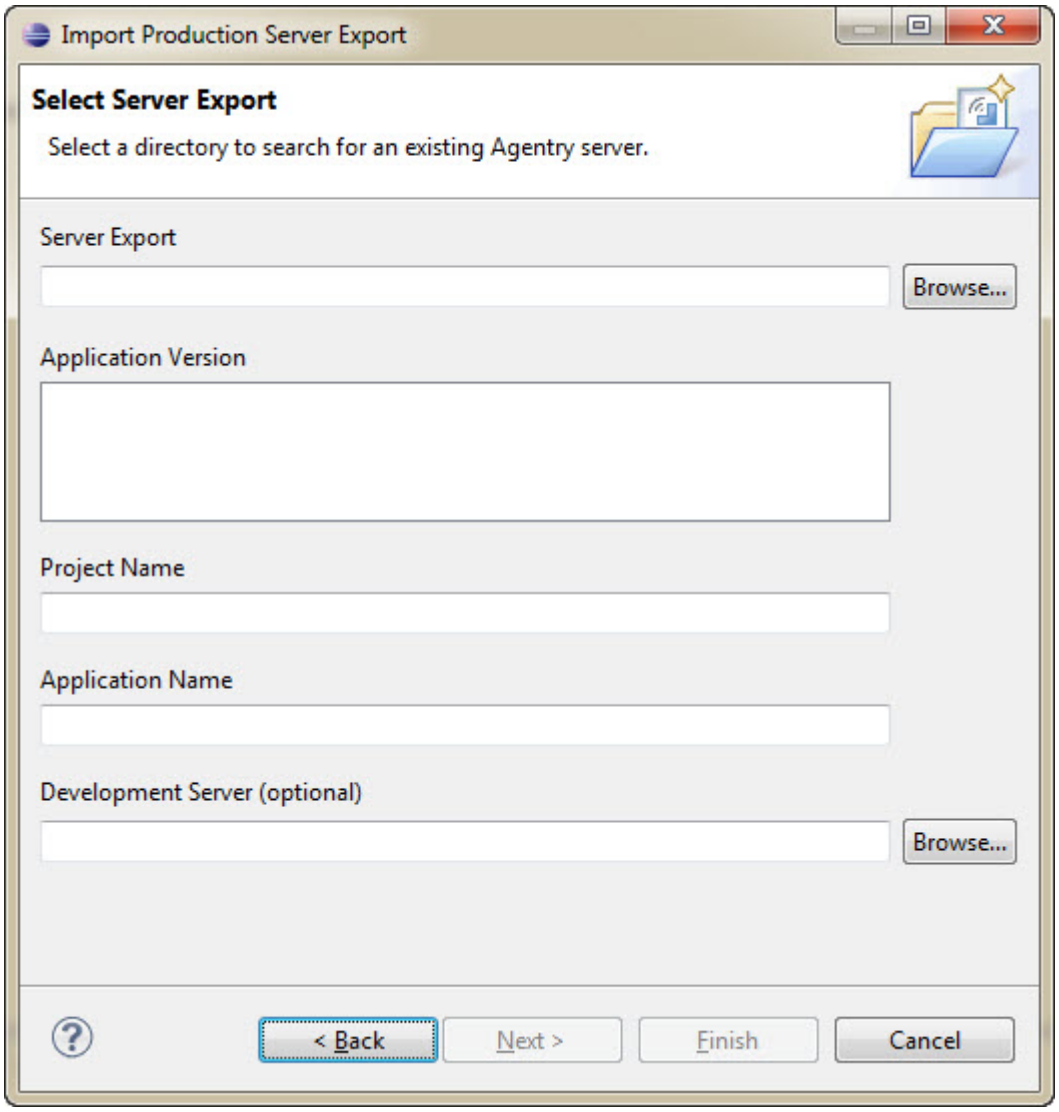
1. If not already open, open or create the Eclipse workspace in which to import the new project. Opening or creating a workspace in Eclipse begins by selecting the menu item *Files* > *Switch Workspace* and following the on-screen instructions.
2. Right-click an empty area in the Project Explorer view and select the menu item *Import*. Alternately, select the menu item *Files* > *Import* in the main menu of Eclipse.

   The Select Import Source screen displays.



3. On this screen, there are the different import sources for Eclipse. Expand the *Agentry Project* folder and select the appropriate source by highlighting it. Click *Next* to continue.

The Select Source screen displays. This screen is slightly different depending on the selected import source type. The following example is for a source type of production server export:



4. In this screen the information entered is dependent on the source type selected in the previous step. Enter the information according to the following:
   a. The specific item selected for the source is different based on the type. Click *Browse* to select the source.
   b. The *Source Application* box is only displayed when the source type is an SAP Mobile Application server. The Source Application box lists the name of the application published to that server instance. If the selected server instance is a production server, each published version currently residing on that server is listed. You can import any one of those versions.
   c. The *Application Name* is the name given to the mobile application that is created by the import. The Application Name is the value of the name attribute within the application definition and can contain no white space. This field is read-only if the selected import source is an Agentry share repository.
   d. The *Project Name* is the name for the project within the Eclipse workspace. The Project Name must be a unique project name for the workspace and white space is allowed.

e. The *Development Server (optional)* is the SAP Mobile Platform for the application project being created. Any script files contained in the source project are copied to this server. By default, this field is set to the server from which the project is imported, if that server instance is a development server. Leave the option set as-is if you are setting up a development server for a new project, or change to a different development server if necessary.

5. Verify the information entered is accurate and complete. Click *Finish* to perform the project import.

   A new project is created by importing the definitions from the selected import source. The project is listed in the Project Explorer view and is automatically opened.

Results

After this process is complete, the new project is added to the Eclipse workspace. The project opens and displays in the Agentry perspective within Eclipse. The application name and project name match those values entered in the Import wizard. If the application project source was created using a previous version of the SAP Mobile Platform, the new project upgrades during the import to the version of the current Agentry.

If the selected import source was an Agentry share repository, the new project contains the definitions found in the top revision of that repository. The revision is checked out to the current user. Subsequent changes made to the application project are tagged with that user ID. The project is connected to the selected share repository and you can update from it. Commit changes made locally to this repository.

5 Installing the SAP Work Manager Client

5.1 Attached Documents Client-Side Security

Any documents that are uploaded using the SAP Work Manager application are stored in the SAP ERP back end. These documents can either be found as an object or as a link in the SAP document management system.

The application does not perform a virus scan of documents while uploading from the local device or downloading them to your local device. It is therefore important to ensure that you use a third party virus scanner to upload and download virus free attachments. An alternative is to implement the SAP Virus Scan Interface (VSI). The VSI allows customers to easily embed an external certified virus scanner of their choice.

See SAP Note 817623: Frequent Questions about VSI in SAP Applications, for which languages and platforms are supported, and where to get further guidance on implementing the SAP VSI.

5.2 Installing the SAP Work Manager Client on Windows CE

Context

The SAP Work Manager client for Windows Mobile requires that the Windows Mobile device is connected to the PC using its docking station or wireless connection. It also requires that ActiveSync or My Mobiler has created a connection with the device.

This client installation application installs the client application created to run on any one of the supported Windows Mobile devices.

To install the SAP Work Manager client:

Procedure

1. Launch the installer executable file from the installation CD.

   The Welcome screen displays.
2. From the Welcome screen, click *Next* to begin installing the SAP Work Manager client.

   The License Agreement screen displays.
3. To continue with the installation, click *I Agree* to accept the license agreement.

   The Choose Components screen displays.

4. From the Choose Components screen, select the options that match your device. If your device is not equipped with a built-in scanner, select *No Scanner*. Likewise, if camera functionality is not to be used, or if the device does not have a camera, select the *No Camera* option. Click *Install* to continue.

   The status screen is displayed, indicating the installation status.
5. The notice screen is displayed, asking where to install the client. When prompted, click *Yes* to select the default for the device.

   The installer installs the client to the target device. When installation is complete, a prompt asks you to check the target device to see if additional steps are necessary to complete the installation.
6. Check the target device for any additional steps. Click *OK* to complete the installation.

   The device requires confirmation for the installation. Actions may vary depending on device.

   The installer installs the SAP Work Manager client. When complete, the Complete screen of the installer displays.
7. Click *Finish* to close the wizard.

5.3 Installing the SAP Work Manager Client on Windows

Context

In addition to the mobile client, there is also a client for the SAP Work Manager client application for Windows systems.

To install the Windows client:

Procedure

1. Start the Windows client installer.
2. Click *Next* on the Welcome screen.
3. Click *Yes* on the License Agreement screen to accept the terms and to continue the installation.
4. In the next screen, select the installation destination for the client, or keep the default location, and then click *Install*.
5. When the installation is complete, click *Finish* to close the wizard.

Results

When installation is complete, the SAP Work Manager client for Windows is installed.

5.4 Installing the SAP Work Manager Client on Apple iOS and Android Devices

Prerequisites

> ⓘ Note
>
> Only iOS 9.x and 10.x devices are supported by the SAP Work Manager client.

The SAP Work Manager client for Apple products requires that the device is connected to a computer with iTunes installed. The mobile device must be able to connect to the iTunes library to download the client application. Android devices must be able to connect to Google Play to download the client application.

Context

Apple iOS: To install the SAP Work Manager client, perform a search in the Apple App Store through the device itself.

If you are not installing through the device using the Internet, plug your device into a desktop or laptop computer with a connection to iTunes. Make sure the computer with iTunes has downloaded a copy of the SAP Work Manager client. When the device is synced with iTunes on the desktop or laptop computer, the SAP Work Manager client is installed on the target client device. All information concerning the specific version and processor type of the client device is determined automatically by the installer.

Android: To install the SAP Work Manager client, perform a search in Google Play through the device itself.

Procedure

1. **Apple iOS:** Open the App Store and search on "SAP Work Manager client" to pull up the most current application and tap *Install*.
2. **Android:** Open Google Play and search on "SAP Work Manager client" to pull up the most current application and tap *Install*.

Results

The SAP Work Manager client is installed on your mobile device.

Next Steps

Synchronize the newly installed SAP Work Manager client with the SAP Mobile Platform containing the published application that is deployed to users. You can synchronize the application, or the intended user of the application can perform this action. The user ID and password information entered during this initial transmit are stored as the credentials of the user for the device going forward. If a different user ID and password is entered, a user change takes place, along with the resulting behavior of that change.

5.5 Backing up the SAP Work Manager Client

You can back up the SAP Inventory Manager client using any method for backing up a mobile installation for the operating system of the mobile device.

Use the method your network administrator recommends.

To fully back up an SAP Work Manager client installation, back up the files located in the installation folder.

To recover a client installation, install the client on the mobile device in the same manner in which it was initially installed. The files copied in backup can then be placed in the installation location on the device.

> ⓘ Note
>
> If the SAP Work Manager client stops running on the client device for any reason, all data is stored in the file, `Agentry.db`, on the device. Restarting the SAP Work Manager client and logging back in displays this data to the user. Recovery of the SAP Work Manager client by reinstalling or other more drastic measures is rarely necessary.

6 Authorizations Required for Mobile Users

For authorization information for SAP Work Manager, see the SAP Work Manager Security Guide, *Required and Essential Authorizations* **topics.**

7 Third Party Authentication

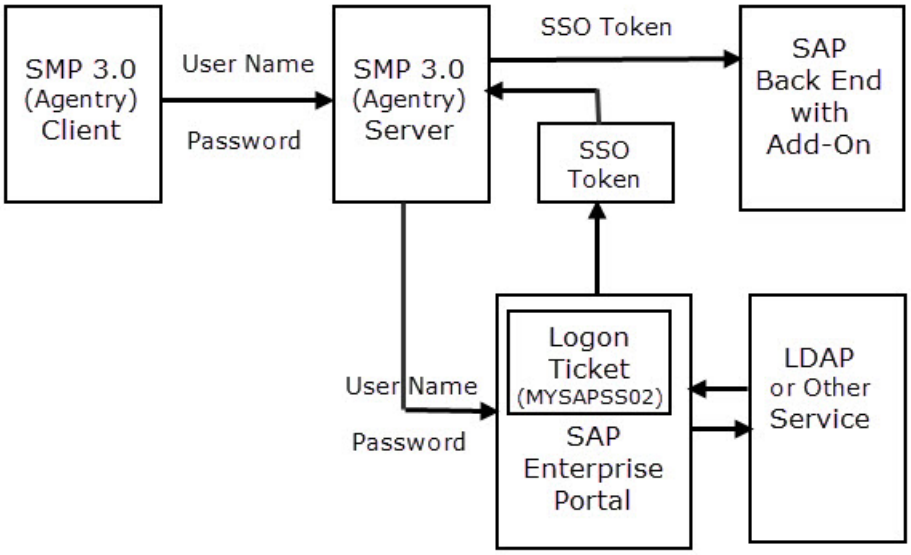
7.1 Configuring SSO and JAAS

Use the following topics to fully configure SSO and JAAS in your mobile application:

- SSO Ticket Authentication and JAAS Custom Login Module Installation [page 44]
- Configuring SSO and JAAS in the JavaBE.ini File [page 40]

Note that the way this is configured is not technically *SSO*. The client device logs into the server using the supplied user name and password for the Agentry client. You supply the user name and password to the client. These get passed on to the server, which then logs on to the Cockpit, and uses these same credentials.

The client does not have the SSO ticket, unlike in a true SSO configuration. The server logs into the ticket portal and the portal issues the SSO cookie. The server then sends the cookie to the SAP back end server as its back end credentials.



7.1.1 Configuring SSO and JAAS in the JavaBE.ini File

Use this section to configure your SSO and JAAS settings in the `JavaBE.ini` file.

SSO Configuration

To enable single-sign on (SSO) authentication on the SAP Mobile Platform, you will need to make configuration changes to the `[USER_AUTH_SSO]` section of the `JavaBE.ini` file. Once all changes are made, be sure to restart the SAP Mobile Platform in order for the modifications to take effect.

Following is a sample configuration `[USER_AUTH_SSO]` section from the `JavaBE.ini` file. Below this example is a list of the available parameters and their descriptions.

Sample Code

```
[USER_AUTH_SSO]
; referenced when LOGON_METHOD=USER_AUTH_SSO
; SSO related information for user in LoginModuleSSO to facilitate Login to
; an SAP system using tickets from a message server.
; PORTAL_URL=https://<server>:<port>/sap/bc/webdynpro
; PORTAL_URL=http://<server>:<port>/irj/portal
; PORTAL_URL=https://localhost/irj/portal/
PORTAL_URL=https://localhost/irj/portal.client/verifier/
; verification file from the portal.
; Not required, but if portal names are different from the authenticated user
name,
; it will be needed to decode the name.
VERIFICATION_USE=true
VERIFICATION_FILENAME=<server>.verify.pse
; VERIFICATION_PASSWORD=rzdrj1<9~gyqdv0?6r}f
; VERIFICATION_PASSWORD_ENCODED=true
; keystore for client authentication to the server, if required.
KEY_STORE_USE=true
; KEY_STORE_TYPE=WINDOWS-MY
; KEY_STORE_TYPE=jks
KEY_STORE_FILENAME=local.client.certificate.keystore
; KEY_STORE_FILENAME=client.p12
; KEY_STORE_PASSWORD=simplepass
; KEY_STORE_PASSWORD_ENCODED=false
; trust store for server authentication to the client (accepting only trusted
servers to prevent MIM attacks)
TRUST_STORE_USE=true
; TRUST_STORE_TYPE=WINDOWS-ROOT TRUST_STORE_TYPE=jks
TRUST_STORE_FILENAME=local.certificate.truststore
TRUST_STORE_PASSWORD=simplepass
; TRUST_STORE_PASSWORD_ENCODED=false
COOKIE=MYSAPSO2
HTTPTYPE=https
SSL_VERSION=SSLv3
JAVA_SECURITY_DEBUG=true
JAVA_NET_DEBUG=true
```

Portal URL

The `PORTAL_URL` parameters tell the server where to log in from and where to get a ticket.

Verification

The `VERIFICATION` parameters means that the ticket does not have to be validated on the SAP Mobile Platform, and this step may be skipped on some systems. If the user name presented to the portal differs from the name returned on the ticket, verification is needed as the new user name is grabbed during that process.

VERIFICATION_USE (true/false): If true, will attempt to verify the ticket returned from the portal using a PSE verification file generated by the portal. The transaction code *STRUST* may also be used to generate this PSE file.

VERIFICATION_FILENAME: The name of the PSE file to use for verification. Path is relative to the application installation directory.

VERIFICATION_PASSWORD: The password to use if the verification file is password protected (i.e., forwarded to the `SAPSSOEXT` libraries).

VERIFICATION_PASSWORD_ENCODED (true/false): Indicates if the password is encoded using the server's `quickPW.exe` command line application so the clear text password is not included in the `JavaBE.ini` file.

Keystore

The `KEYSTORE` parameters are required if the SSL connection to the portal server requires a client certificate to authenticate the client. Many server setups to not require a client certificate, so this is not necessary in those situations. Here, we pass in many options used for key management in Java.

KEY_STORE_USE (true/false): Indicates if the client certificate keystore should be provided to the `SSLContext` class used to initiate the connection.

KEY_STORE_TYPE: The type of keystore to use. This value is passed into the `KeyStore.getInstance(_)` function. On Windows machines, you can choose the default keystore for the user indicated by the value `WINDOWS-MY`.

KEY_STORE_FILENAME: The file name of the keystore to load. The path is relative to the application's installation directory.

KEY_STORE_PASSWORD: The password of the keystore.

KEY_STORE_PASSWORD_ENCODED (true/false): Indicates if the password is encoded using the server's `quickPW.exe` command line application so the clear text password is not included in the `JavaBE.ini` file.

`TRUST_STORE` parameters are required to create an SSL connection to a portal server. The trust store should contain the server certificates required to achieve a chain of trust to the server (ensuring that the server you connect to is the one you intended to connect to and not an imposter, as in a 'monkey in the middle' attack). These options are the same as the parameters for `KEY_STORE`.

Note that a truststore is a keystore, only a truststore typically contains the chain of authority certificates and not private keys.

Custom JAAS Configuration

To enable Java Authentication and Authorization Service (JAAS) authentication on the SAP Mobile Platform, you will need to make configuration changes to the `[USER_AUTH_CUSTOM]` section of the `JavaBE.ini` file. Once all changes are made, be sure to restart the SAP Mobile Platform in order for the modifications to take effect.

Configuration contains three things per module class: the class itself, its flag (`REQUIRED`, `REQUISITE`, `SUFFICIENT`, or `OPTIONAL`), and the options to pass into the module.

Following is a sample configuration `[USER_AUTH_CUSTOM]` section from the `JavaBE.ini` file. Below this example is a list of the available parameters and their descriptions.

Sample Code

```
[USER_AUTH_CUSTOM]
; referenced when
```

```
; LOGON_METHOD=USER_AUTH_CUSTOM
; custom defined log in configuration that uses JAAS for authentication
MODULE_CLASS_1=<server>
MODULE_CLASS_1_FLAG=REQUIRED
MODULE_CLASS_1_OPTION_1_KEY=CLIENT_NUM
MODULE_CLASS_1_OPTION_1_VALUE=800
MODULE_CLASS_1_OPTION_2_KEY=HOST
MODULE_CLASS_1_OPTION_2_VALUE=<server>
MODULE_CLASS_1_OPTION_3_KEY=SYS_NUM
MODULE_CLASS_1_OPTION_3_VALUE=11

; MODULE_CLASS_2=
; MODULE_CLASS_2_FLAG=
; MODULE_CLASS_3=
; MODULE_CLASS_3_FLAG=
; MODULE_CLASS_[1...n]: The module class to load.
; The number indicates the order they are loaded in and processed, starting
with 1,
; and moving on until it does not find the next higher integer number.
; MODULE_CLASS_[1...n]_FLAG: The flag for the indicated module.
; MODULE_CLASS_[1...n]_OPTION_[1...m]_KEY: The key of a key/value pair for an
option to load
; in for the indicated module. The numbers indicate the module to pass the
options to and the
; order of options to pass into the given module. They are loaded in order,
starting with 1
; and moving forward until it does not find the higher integer number.
; MODULE_CLASS_[1...n]_OPTION_[1...m]_VALUE: The value of a key/value pair
for an option
; to load in for the indicated module.
```

Log On Method Configuration

To enable the two added options for log on methods on the SAP Mobile Platform, you will need to make configuration changes to the `[LOGON_METHOD]` section of the `JavaBE.ini` file. Once all changes are made, be sure to restart the SAP Mobile Platform in order for the modifications to take effect.

Following is a sample configuration `[LOGON_METHOD]` section from the `JavaBE.ini` file. Below this example is a list of the available parameters and their descriptions.

Sample Code

```
[LOGON_METHOD]
; USER_AUTH if standard UID/Password authentication is used
; USER_AUTH_GLOBAL if pooled connections using single UID/password is used
; USER_AUTH_GROUP if UID/password authentication with SAP Message Server Load
Balancing is used
; USER_AUTH_SSO if SSO2 ticket authentication with SAP Message Server
; USER_AUTH_CUSTOM for a custom log in module setup
LOGON_METHOD=USER_AUTH_SSO
; USER_AUTH_SSO: Uses the SSO module and passes in options provided
; in the [USER_AUTH_SSO] section of the JavaBE.ini file.
; USER_AUTH_CUSTOM: Uses the generic JAAS handling for authentication by
; instantiating the modules and options as indicated in the
; [USER_AUTH_CUSTOM] section of the JavaBE.ini file.
```

Additional Reference Information

Keystores / Truststores

Through convention we call the file containing trusted public certificates of web servers a 'truststore'. Its underlying structure is a 'keystore', so for our application purposes, they are referred to as keystores. Java allows for customization of keystores. There is information on the key tool application provided by the JRE and JDK here:

http://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html

The above link details how to import certificates from the web server, generating certificate signing requests (CSRs) and more. The default keystore type is `JKS` and the default store password is `changeit`. These are useful to know when creating a keystore.

It may also be useful to have OpenSSL installed for use of other key formats.

Keystore Types

The following URL provides information on keystore types:

http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#KeyStore

SSL Context Algorithms

The following URL provides information on SSL context algorithms:

http://docs.oracle.com/javase/6/docs/technotes/guides/security/StandardNames.html#SSLContext

http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#SSLContext

Callback Handlers

The following URL provides documentation on configuration options:

http://docs.oracle.com/javase/6/docs/technotes/guides/security/jgss/tutorials/LoginConfigFile.html

The implementation in the mobile applications discussed in these *Installation Guides* use a different syntax, but the parameters are provided to the configuration all the same.

7.1.2 Configuring the SSO Ticket Authentication

Prerequisites

Ensure the following two `.dll` libraries are located within the SAP Mobile Platform installation directory. If they were not copied to the SAP Mobile Platform installation directory during the SAP Mobile Platform installation through the installer, copy them now:

- `SAPSSOEXT.DLL`
- `SAPSECU.DLL`

These libraries are needed for SSO ticket verification.

Context

Use the following procedure to configure SSO ticket authentication. See the topic Configuring SSO and JAAS in the JavaBE.ini File [page 40] for detailed descriptions on all the parameters.

Procedure

1. Open the `JavaBE.ini` file in your text editor of choice and navigate to the `[LOGON_METHOD]` section.
2. Set the `LOGON_METHOD` parameter to `USER_AUTH_SSO`, as shown in the step result.

   Sample Code

   ```
   [LOGON_METHOD]
   LOGON_METHOD=USER_AUTH_SSO
   ```

3. Navigate to the `[USER_AUTH_SSO]` section for *Steps 4–11*.
4. Set the `PORTAL_URL` parameter to the http or https of your SAP NetWeaver gateway portal server. The portal URL is the URL that the module uses to log in and retrieve the SSO2 token.

   Sample Code

   ```
   PORTAL_URL=https://<server>:<port>/irj/portal
   ```

5. If you decide to verify tickets on the server, you can set the `VERFICATION_USE` parameter to `true`. Generate a verification PSE file from the SAP GUI on the SAP server (transaction code *STRUST* or *STRUSTSSO2*) and enter the file name in the `VERIFICATION_FILENAME` parameter. This is the file name relative to the installation directory of your server. If you choose to have the file password-protected, enter the password here as well. If you opt to have the password encoded, set the `VERIFICATION_PASSWORD_ENCODED` parameter to *true*.

   If there is a chance the user name of the back-end server differs from the user name of the SAP server, set the verification file. Otherwise, this step is optional.

   Sample Code

   ```
   VERIFICATION_USE=false
   VERIFICATION_FILENAME=verify.pse
   VERIFICATION_PASSWORD=1234
   VERIFICATION_PASSWORD_ENCODED=false
   ```

   Note

   If you do not want clear text passwords in your configuration files, use the QUICKPW.exe application provided with the installer.

6. To set up a keystore or truststore, set its respective `_USE` flag to *true*, supply the `_TYPE` parameter, and give the `_FILENAME` as a relative path to the server installation directory. Fill in the `_PASSWORD` parameter. Similarly to *Step 5*, if you do not want clear text passwords in the `JavaBE.ini` file, use the `QUICKPW.exe` tool to encode it. Be sure to set the `_PASSWORD_ENCODED` flag to *true* if you do encode the password.

Keystore and truststore contain the same options and are very similar. The keystore is used if your SAP server requires client authentication for its SSL implementation (i.e., if the SAP NetWeaver gateway portal server requires any https connections to provide credentials to the web server). The truststore is used by the SSL client to verify that the SAP NetWeaver gateway portal server connected to is the one intended to connect to (it operates identically to the list of trusted CA certificates loaded in your typical web browser). The truststore is required if your SAP NetWeaver gateway portal server is running over SSL (https), while the keystore is optional, depending on the setup of your SAP NetWeaver gateway portal server.

On Windows machines, there are default key/truststores, which you can choose to use with certain keywords as the `_TYPE` parameter. The system keystore of the user has a type of WINDOWS-MY and the default truststore of the system is WINDOWS-ROOT. If you have valid certificates installed on the SAP Mobile Platform server for viewing the SAP NetWeaver gateway portal server, it is possible that using the existing Windows key/truststores is sufficient.

Sample Code

```
KEY_STORE_USE=false
KEY_STORE_FILENAME=keystoreFileName
KEY_STORE_PASSWORD=1234
KEY_STORE_PASSWORD_ENCODED=false
TRUST_STORE_USE=true
TRUST_STORE_TYPE=WINDOWS-ROOT
TRUST_STORE_FILENAME=truststoreFileName
TRUST_STORE_PASSWORD=1234
TRUST_STORE_PASWORD_ENCODED=false
```

Note

More details on creating keystores/truststores can be found in the topic Configuring SSO and JAAS in the JavaBE.ini File [page 40].

7. The `COOKIE` parameter is the name of the cookie to examine for the SSO2 ticket in the response of the server. This parameter is `MYSAPSSO2` by default.

   Sample Code

   ```
   COOKIE=MYSAPSS02
   ```

8. `HTTPTYPE` should reflect the `PORTAL_URL` (i.e., is it a secure connection or not: `http` or `https`).

   Sample Code

   ```
   HTTPTYPE=https
   ```

9. The `SSL_VERSION` parameter lets you customize the version of SSL/TLS for Java to use.

   Sample Code

   ```
   SSL_VERSION=SSLv3
   ```

10. The `JAVA_SECURITY_DEBUG` and `JAVA_NET_DEBUG` parameters set system flags that provide more details about the logon procedure within the Java code to the console window of the Agentry application. These options may help if there are connectivity issues, but should be left as `false` otherwise.

    ```
    JAVA_SECURITY_DEBUG=falseJAVA_NET_DEBUG=false
    ```

11. If end users want to customize their own `SSOClient` and/or `CallbackHandler` classes, they can provide subclassed versions of the provided classes and denote them in the `SSOCLIENT_CLASS` and `CALLBACK_HANDLER_CLASS` parameters.

> **Note**
>
> While the supplied implementation works for most default installations, some end users may have more specific requirements or customizations to add, and can do so using these parameters.

> **Sample Code**

```
SSOCLIENT_CLASS=com.mysap.sso.SSOClient
CALLBACK_HANDLER_CLASS=com.syclo.sap.auth.CallbackHandler
```

Results

The SSO ticket authentication configuration is complete.

Next Steps

If the server is running, restart the SAP Mobile Platform server so the changes to the `JavaBE.ini` file can take effect. Once the server is running with the changes in place, log in with a valid user. If that logon succeeds, the configuration and installation was successful. If an exception is thrown, troubleshoot the configuration and installation parameters.

7.1.3 Configuring the Custom JAAS Logon Module

Context

Use the following procedure to configure the custom JAAS logon module. See the topic Configuring SSO and JAAS in the JavaBE.ini File [page 40] for detailed descriptions on all the parameters, as well as additional links for more information.

Procedure

1. Navigate to the `[LOGON_METHOD]` section in the `JavaBE.ini` file.
2. Set the `LOGON_METHOD` parameter to `USER_AUTH_CUSTOM`.

Sample Code

```
LOGON_METHOD=USER_AUTH_CUSTOM
```

3. Navigate to the `[USER_AUTH_CUSTOM]` section in the `JavaBE.ini` file.
4. The options here follow typical JAAS configuration parameters. You can supply the modules, their flag (REQUIRED/REQUISITE/SUFFICIENT/OPTIONAL) and their options as key/value pairs.

Note

For more information, see the topic Configuring SSO and JAAS in the JavaBE.ini File [page 40].

Sample Code

```
MODULE_CLASS_1=com.syclo.sap.auth.LoginModuleBasic
MODULE_CLASS_1_FLAG=REQUIRED
MODULE_CLASS_1_OPTION_1_KEY=CLIENT_NUM
MODULE_CLASS_1_OPTION_1_VALUE=clientNum
MODULE_CLASS_1_OPTION_2_KEY=HOST
MODULE_CLASS_1_OPTION_2_VALUE=serverHostName
MODULE_CLASS_1_OPTION_3_KEY=SYS_NUM
MODULE_CLASS_1_OPTION_3_VALUE=sysNum
```

5. Similar to *Step 11* in the Configuring the SSO Ticket Authentication [page 44] procedure, if users have a custom callback handler they want to use, they can supply the class name in the `CALLBACK_HANDLER_CLASS` parameter and it will be instantiated.

Sample Code

```
CALLBACK_HANDLER_CLASS=com.syclo.sap.auth.CallbackHandler
```

Results

The JAAS custom module logon installation is complete.

Next Steps

If the server is running, restart the SAP Mobile Platform server so the changes to the `JavaBE.ini` file can take effect. Once the server is running with the changes in place, log in with a valid user. If that logon succeeds, the configuration and installation was successful. If an exception is thrown, troubleshoot the configuration and installation parameters.

7.1.4 Configuring SSO Log In Tickets Generated from the SAP Mobile Platform Server

The public certificate used by the SAP Mobile Platform to generate the SSO tickets must be added in the STRUSTSSO2 transaction as part of the system PSE.

Prerequisites

- The SAP Mobile Platform server is already installed and configured.
- SAP Work Manager is already configured to connect to your SAP back-end system.
- OpenSSL is installed in order to create the necessary certificate.
- You have the appropriate log in accounts to SAP and the SAP Mobile Platform Management Cockpit to make the changes in the procedure below.

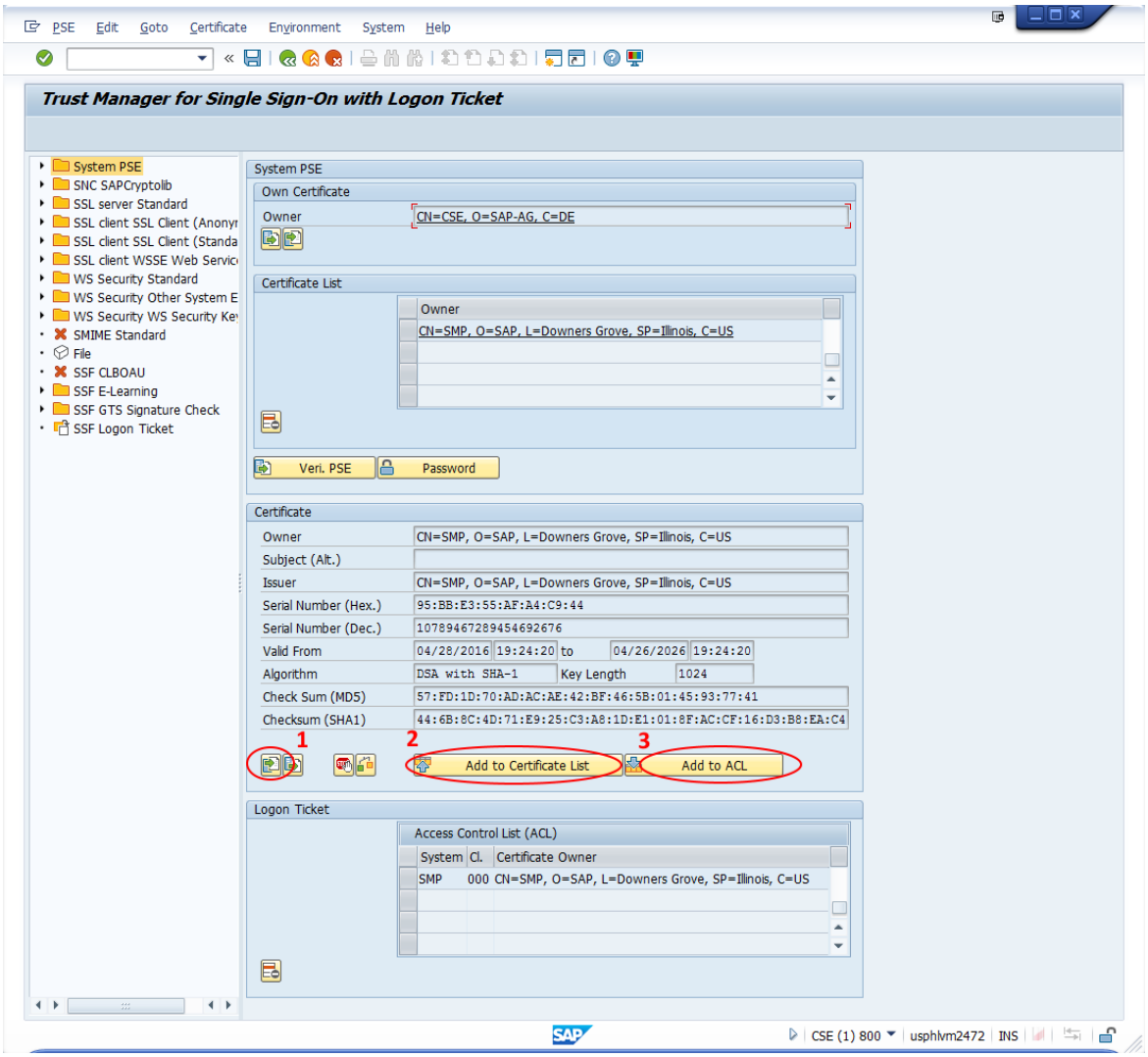
Procedure

1. Create the necessary certificate files for certificate authentication between the SAP Mobile Platform and the SAP back-end system. See below for an example.

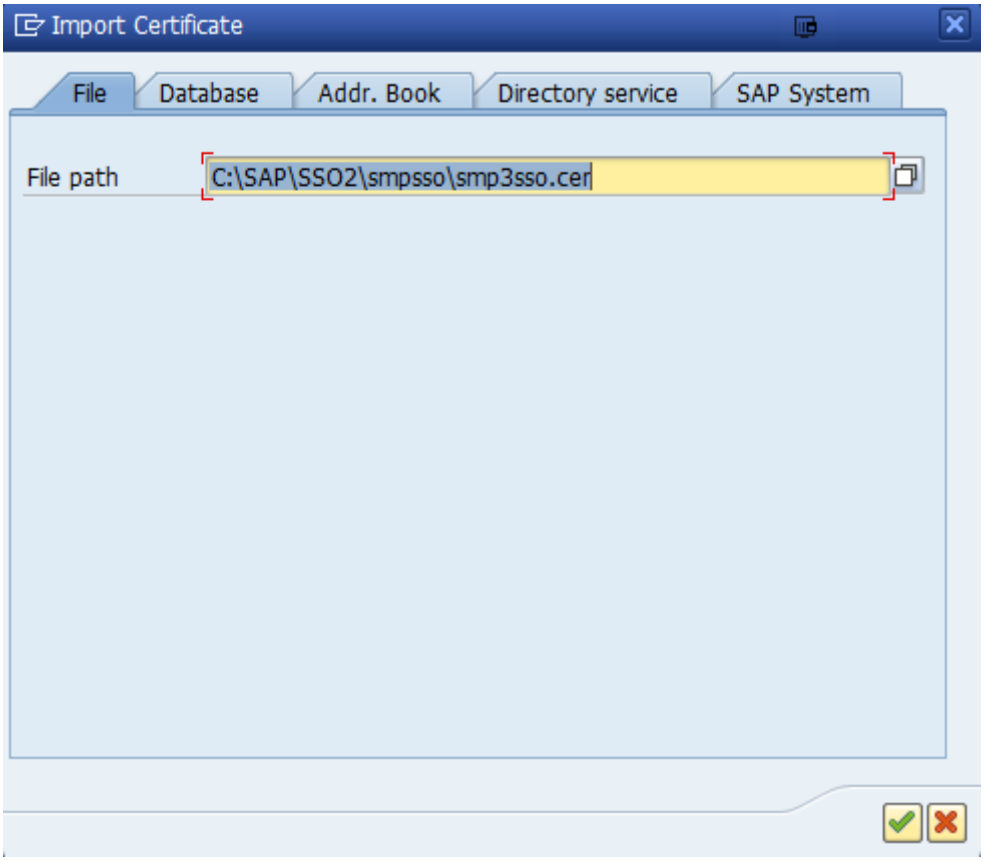
```
C:\SAP\SSO2\smpsso>path=c:\OpenSSL-Win64\bin;c:\OpenSSL-Win64;%path%
C:\SAP\SSO2\smpsso>openssl dsaparam -out dsaparam.pem 1024
Loading 'screen' into random state - done
Generating DSA parameters, 1024 bit long prime
This could take some time
unable to write 'random state'
C:\SAP\SSO2\smpsso>openssl gendsa -out smp3sso.pem dsaparam.pem
Loading 'screen' into random state - done
Generating DSA key, 1024 bits
unable to write 'random state'
C:\SAP\SSO2\smpsso>openssl req -x509 -days 3650 -new -key smp3sso.pem -sha1
-out smp3sso.cer
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Illinois
Locality Name (eg, city) []:Downers Grove
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SAP
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:SMP
Email Address []:
C:\SAP\SSO2\smpsso>openssl pkcs12 -export -in smp3sso.cer -name smp3sso
-inkey smp3sso.pem -out smp3sso.p12
Loading 'screen' into random state - done
Enter Export Password:{password}
Verifying - Enter Export Password:{password}
unable to write 'random state'
C:\SAP\SSO2\smpsso>dir
```

```
    Volume in drive C is OSDisk
    Volume Serial Number is E444-F296
    Directory of C:\SAP\SSO2\smpsso
    04/28/2016  02:24 PM    <DIR>          .
    04/28/2016  02:24 PM    <DIR>          ..
    04/28/2016  02:23 PM               455 dsaparam.pem
    04/28/2016  02:24 PM             1,172 smp3sso.cer
    04/28/2016  02:24 PM             1,596 smp3sso.p12
    04/28/2016  02:23 PM               668 smp3sso.pem
        4 File(s)          3,891 bytes
        2 Dir(s)   8,606,961,664 bytes free
    C:\SAP\SSO2\smpsso>
```

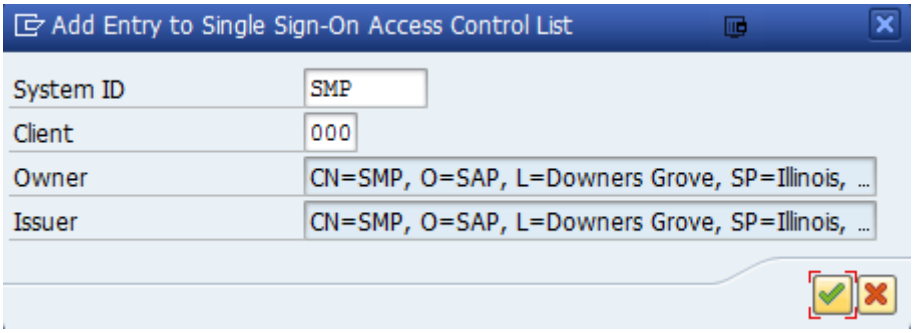
2. Install the `smp3sso.cer` on the SAP back-end system using transaction STRUSTSSO2. You have three choices after accessing the Trust Manager for Single Sign-On with Logon Ticket screen:



- Import the certificate (option 1 in Step 2's screenshot)



- Add to the certificate list (option 2 in Step 2's screenshot): Click the *Add to Certificate* button to move the entry to the Certificate List.
- Add to ACL (option 3 in Step 2's screenshot): Enter the values for the *System ID* and *Client*.
  - System ID: SMP
  - Client: 000

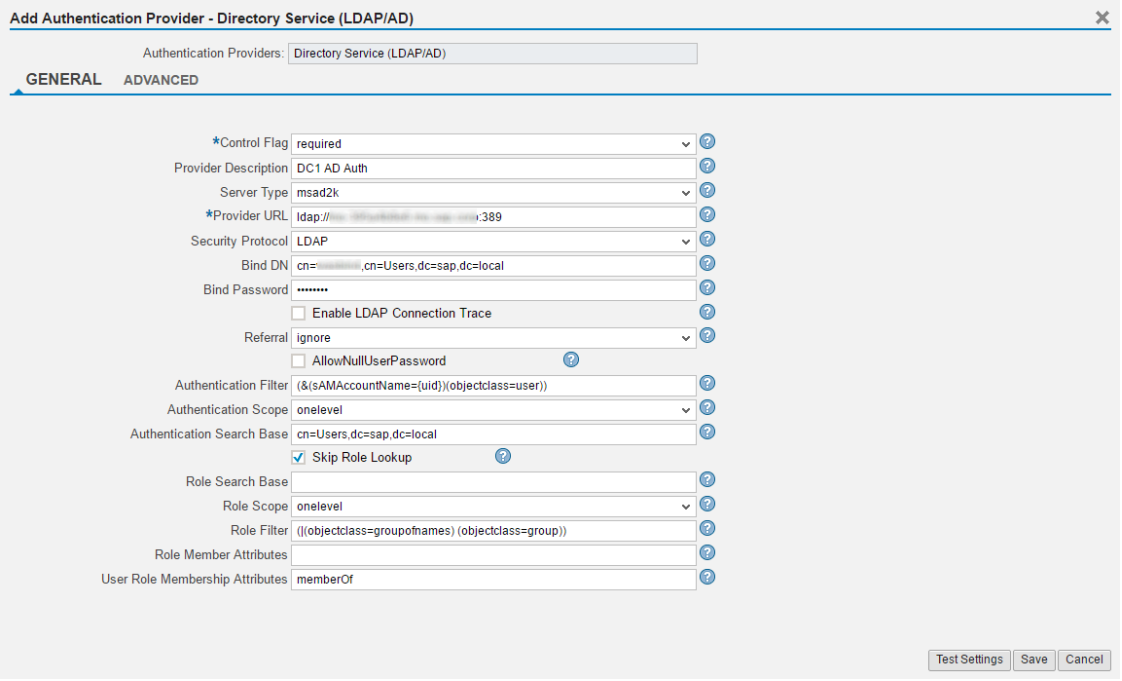


After making your choice, be sure to save your changes.

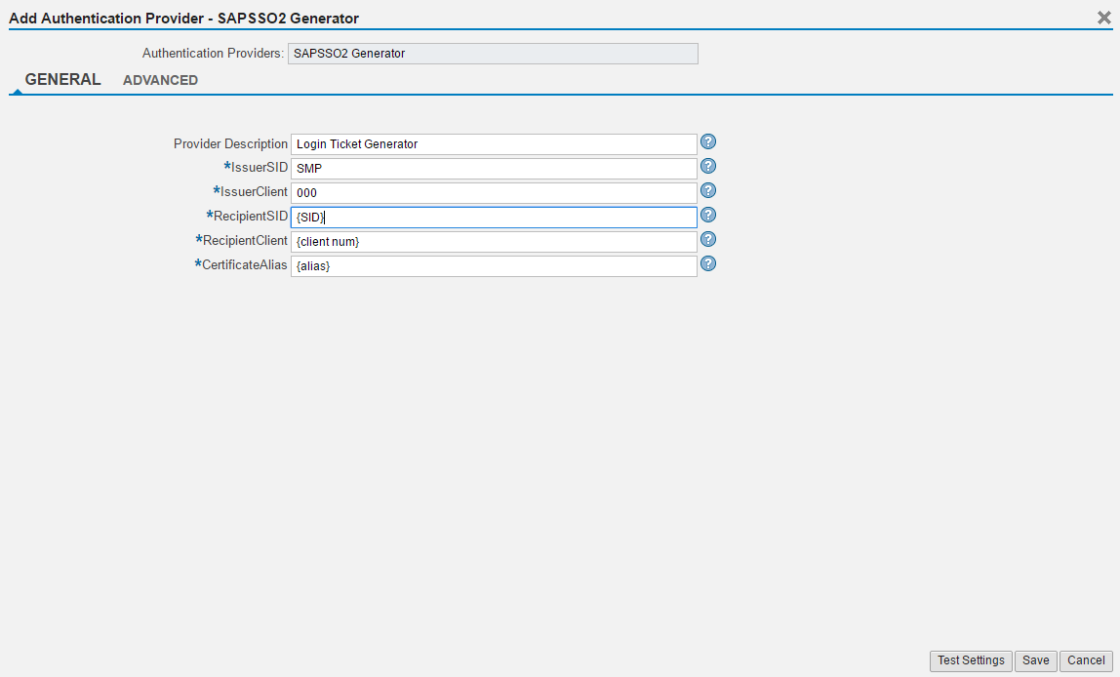
3. Import the `.p12` file on the SAP Mobile Platform server:
   a. Log in to the SAP Mobile Platform Management Cockpit, if not already logged in.
   b. Navigate to *Settings* > *Certificates* > *Shared Keystore Entries* and click the *Import* button.
   c. Input the following details as shown:
      - **Certificate Type:** Change the type to *PKCS#12*
      - **Alias:** Enter an alias to identify the certificate, i.e., *smp3sso*

- **Certificate File:** Your `.p12` file from Step 1
- **Private Key Password:** The password for the `.p12` file from Step 1

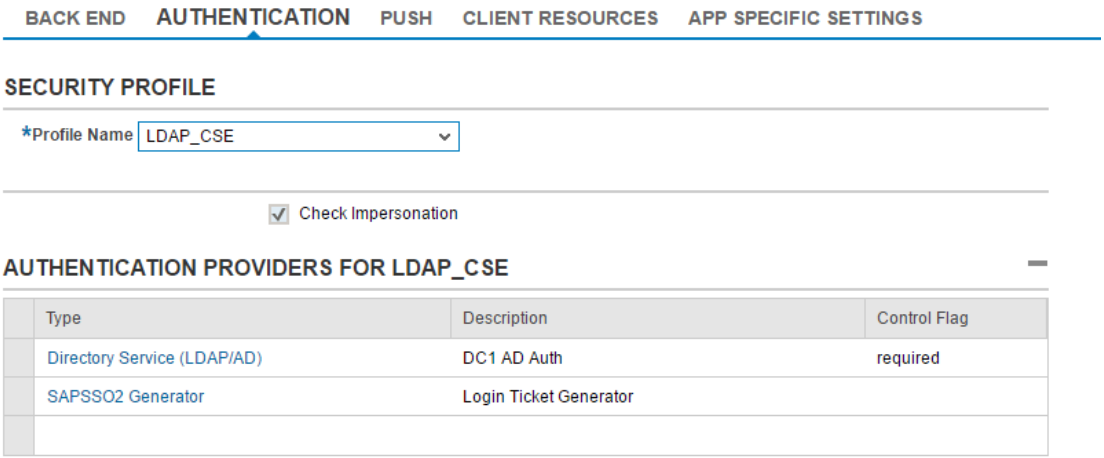
4. Create the new security profile to authenticate.
   a. Navigate to *Settings* > *Security Profiles* and click *New*.
   b. Set the *Name*. This can be anything but should be indicative of what type of authentication it is performing (i.e., *LDAP_[SID]*, where [SID] matches the SID of the SAP system you are setting the SSO against.
   c. Click *Add* and select *Set Directory Service (LDAP/AD)*
   d. Make the following changes to the fields as per the following examples:
      - **Server Type:** msad2k
      - **Provider URL:** ldap://[your LDAP host]:389
      - **Bind DN:** Full DN of the user used for lookup authentication
      - **Bind Password:** [Password for your bind user referenced in the Bind DN]
      - **Authentication Filter:** (&(sAMAccountName={uid})(objectclass=user))
      - **Authentication Scope:** May need to switch to subtree depending on your LDAP setup
      - **Authentication Search Base:** The base reference for your users
      - **Skip Role Lookup:** Checked



   e. Click *Add* and select *SAPSSO2 Generator*.
   f. Configure based on your environment and certificate. See the following for examples:
      - **Provider Description:** Optional description of this provider (i.e., Logon Ticket Generator)
      - **IssuerSID:** SMP
      - **IssuerClient:** 000
      - **RecipientSID:** The SID of the SAP back-end system you are connecting to
      - **RecipientClient:** The client number within the SAP back-end system
      - **CertificateAlias:** [alias from the `.p12` import]



5. Change your SAP Work Manager to use the new authentication scheme.
   a. Change the *Security Profile* for your SAP Work Manager instance to use the new profile. See the screenshot below for an example.



   b. Save your changes.
6. Configure SAP Work Manager to accept the ticket from SAP Mobile Platform.
   a. Copy the `smp3ssogen.jar` file to your application Java directory.
   b. In the SAP Mobile Platform Cockpit, under *Back End* > *JAVA-1*, add the `smp3ssogen.jar` file to the *Class Path*, as follows:

```
•    ./Java;./Java/smp3ssogen.jar;./Java/workmanager-6.3.0.0.jar;./Java/
common-20151103.jar;./sapsso.jar;./sapjco3.jar;./ini4j.jar;
```

c. Edit the `JavaBE.ini` to set the `[LOGON_METHOD]` entries. See the following for an example:

```
[LOGON_METHOD]
;  USER_AUTH if standard UID/Password authentication is used
;  USER_AUTH_GLOBAL if pooled connections using single UID/Password is used
;  USER_AUTH_GROUP if UID/Password authentication with SAP Message Server
;   (load balancing) is used
;  USER_AUTH_SSO if SSO2 ticket authentication with SAP Portal Server is
used
;  USER_AUTH_CUSTOM for a custom logon module setup
LOGON_METHOD=USER_AUTH_CUSTOM
SERVICE_USER_LOGON_METHOD=USER_AUTH
PUSH_USER_LOGON_METHOD=USER_AUTH
```

d. Edit the `JavaBE.ini` to set the `[USER_AUTH_CUSTOM]` entries. See the following for an example:

```
[USER_AUTH_CUSTOM]
;    referenced when LOGON_METHOD=USER_AUTH_CUSTOM
;    custom defined logon configuration that use JAAS for authentication
;
MODULE_CLASS_1=com.syclo.sap.auth.smpssogen.LoginModuleSMPSSO2
MODULE_CLASS_1_FLAG=REQUIRED
MODULE_CLASS_1_OPTION_1_KEY=CLIENT_NUM
MODULE_CLASS_1_OPTION_1_VALUE={client num}
MODULE_CLASS_1_OPTION_2_KEY=HOST
MODULE_CLASS_1_OPTION_2_VALUE={your ECC hostname}
MODULE_CLASS_1_OPTION_3_KEY=SYS_NUM
MODULE_CLASS_1_OPTION_3_VALUE={system num}
; class to instantiate for the callback handler
CALLBACK_HANDLER_CLASS=com.syclo.sap.auth.smpssogen.CallbackHandler
BYPASS_USERID_CHECK=true
```

e. Save the `JaveBE.ini` file.

7. Restart your SAP Work Manager application in the SAP Mobile Platform Management Cockpit.
8. Test connecting from your SAP Work Manager client.

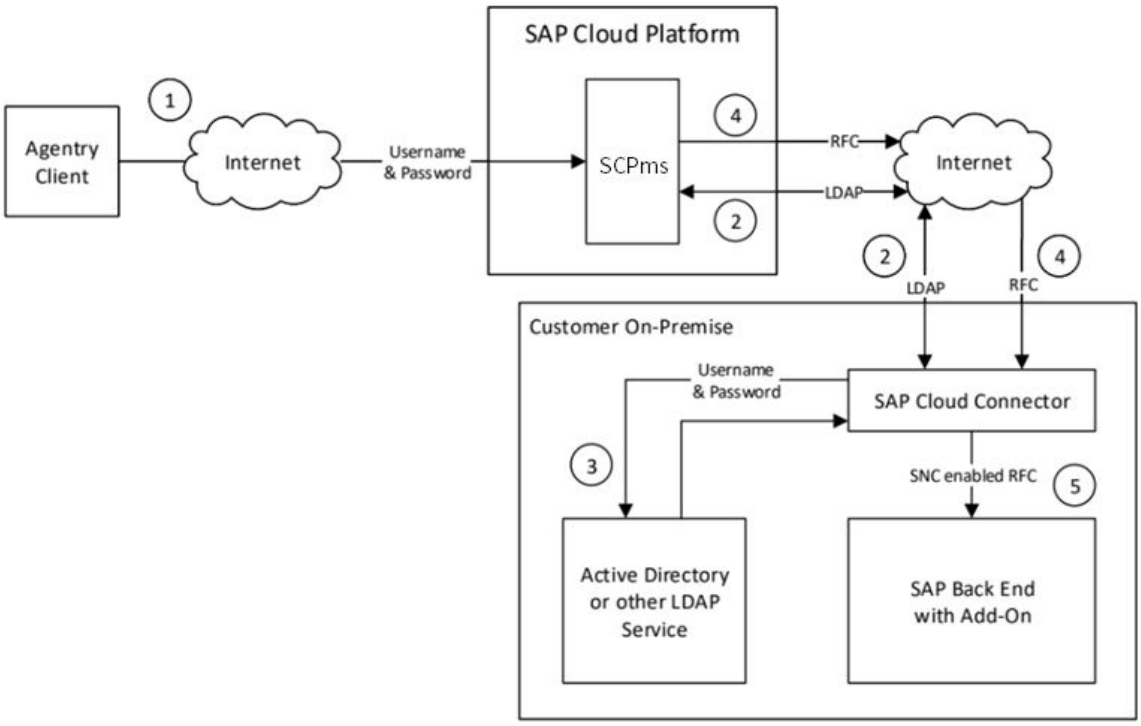
7.2 Principal Propagation

Use

Principle propagation allows destinations to forward the identity of an on-demand user to the SAP Cloud Connector, which then forwards it to the back-end system of the relevant on-premise system. An on-demand user does not need to provide their identity for each connection to an on-premise system when using the Cloud Connector. The full identity is propagated.

Principal Propagation Overview

The following is a diagram and the high-level steps that occur when an LDAP/AD user authenticates through SAP Business Technology Platform Mobile Services:



1. User enters their LDAP/AD user name and password on the startup of the SAP Work Manager client application.
2. SAP Business Technology Platform Mobile Services uses basic authentication to validate the user name and password.
3. An on-premise user store is used to authenticate the users.
4. SAP Work Manager uses principal propagation to connect to the back-end system.
5. The connection to the Cloud Connector uses an SNC enabled RFC connection.

7.2.1 Enabling Support for Principal Propagation

Prerequisites

- Principal propagation requires Mobile Add-On for ERP 6.2.0, version SP05 or above.
- The back end SAP system is configured to allow for principal propagation through SNC secured RFC communications. For more information, see the procedure Configure Principal Propagation to an ABAP System for RFC.

- Mobile client is using Agentry client version 3.1 or Agentry client version 3.0 SP16 PL08

Context

To enable support for principal propagation, changes are needed in the `JavaBE.ini` file, as well as the changes required to support the SAP Work Manager application set up on the SAP S/4HANA cloud environment. Note that most of the cloud environment configuration is performed by your system installer.

Procedure

1. Open the `JavaBE.ini` file in the text editor of your choice and navigate to the `[JCO]` section.
2. Set `CONNECTION_TYPE` to `PRINCIPAL_PROPAGATION`.

   Sample Code

   ```
   [JCO]
   CLASS=JCO3
   CONNECTION_TYPE=PRINCIPAL_PROPAGATION
   ```

3. Ensure that the `LOGON_METHOD` in the `[LOGON_METHOD]` section is set to `USER_AUTH`. Make any changes to the type of logon in the configured destinations.

   Sample Code

   ```
   [LOGON_METHOD]
   LOGON_METHOD=USER_AUTH
   ```

4. If custom destinations are use, specify the customized destination names. Navigate to the `[JCO3]` section and input your desired custom destination names into the provided templates:

   Sample Code

   ```
   [JCO3]
   ;customized destination names
   DESTINATION_SERVICE_NAME=CUSTOM_DESTINATION_SERVICE
   DESTINATION_PUSH_NAME=CUSTOM_DESTINATION_PUSH
   DESTINATION_SESSION_NAME=CUSTOM_DESTINATION_SESSION
   ```

5. Navigate to the `[CLOUD]` section.
6. Set the `CREDENTIAL_SETTING_TIMEOUT` to the same setting found on the SAP Business Technology Platform Mobile Services (SCPms) Cockpit.

   Set the timeout to longer than what you expect your longest transmit will take, as credentials cannot refresh midtransit.

   Note

   Remember to use the same setting on the SAP BTP services Cockpit.

Sample Code

```
[CLOUD]
.
.
.
;timeout for credentials in seconds; should match what is in the SCPms
Cockpit for timeout
;or in a published Agentry.ini file if one is used.
CREDENTIAL_SESSION_TIMEOUT=1200
```

7. Still in the `[CLOUD]` section, add the following configuration option:

Sample Code

```
;keep destination thread open until timeout is reached rather than close
after each backend session
FORCE_KEEP_DESTINATION_THREAD_ALIVE=TRUE
```

8. Navigate to the `[USER_AUTH]` section.
9. Bypass the user ID check by setting it to `TRUE`.

Sample Code

```
[USER_AUTH]
BYPASS_USERID_CHECK=TRUE
```

10. **Optional:** Still in the `[USER_AUTH]` section, set the following configuration option:

Sample Code

```
ALLOW_USERNAME_REMAPPING=true
```

Out of the box, the configuration option is set to `FALSE`. Set it to `TRUE` to allow the remapping of a username when an SSO username does not match with a back end username. Principal propagation, and SSO in some cases, does not know the back end username at the time of authentication, and must remap it after authentication.

11. Save your changes.

Next Steps

Republish the SAP Work Manager application in order for the modifications to take effect.

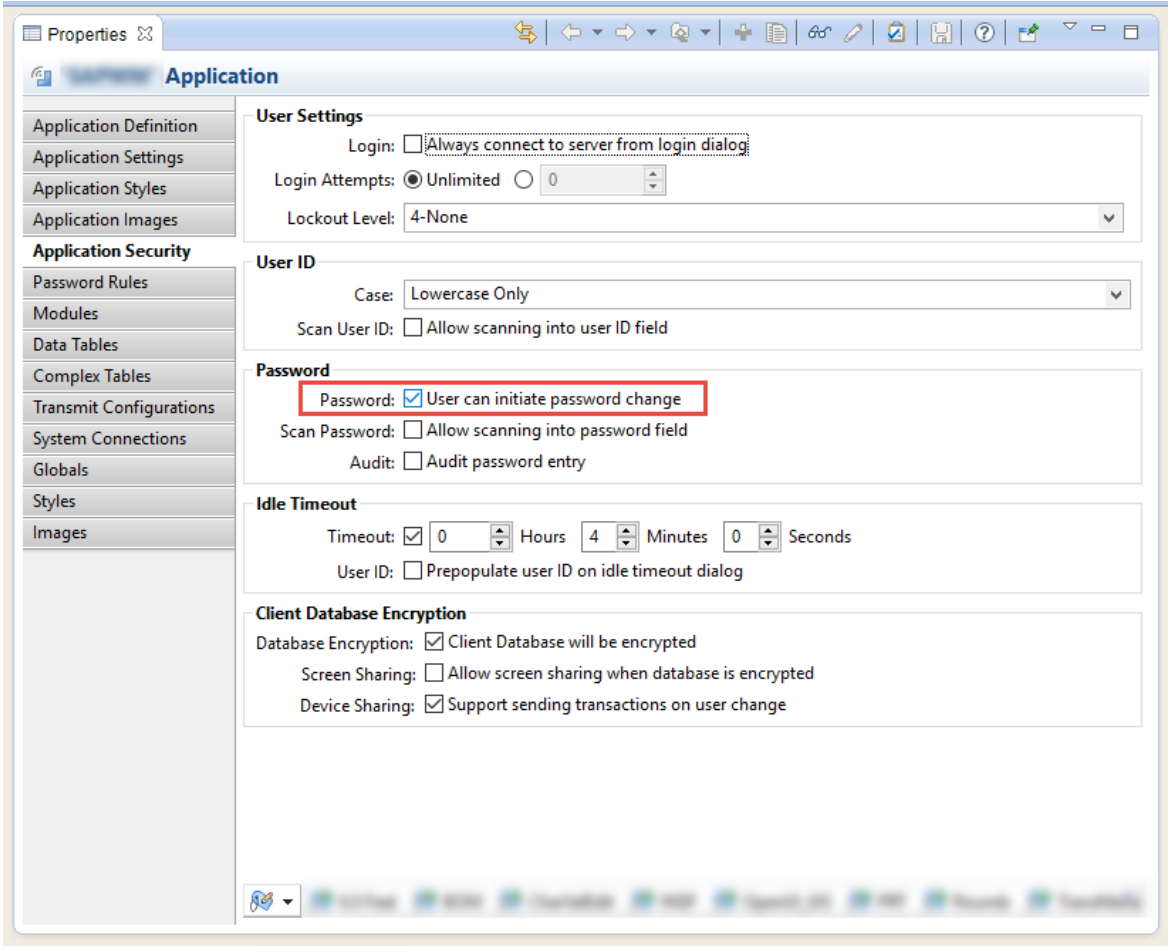
7.2.2 Turning Off Password Change Functionality

Context

If you use a third party identity provider, as is typically done with SSO and principal propagation authentication mechanisms, the password change functionality does not work. Turn off the password change function through the Agentry as follows:

Procedure

1. Select *Application* from the tree in *Project Explorer*.
2. Select the *Application Security* tab.
3. Uncheck the box next to *User can initiate password change*.



4. Click the *Save* button.
5. Publish the application and deploy it to your server.

8 Logging

Logs contain structured data about the user and their transmit in each entry by default.

The following sections detail information about the logging feature.

Logger Class Outputs

Logging functionality is based on the log4j 2.x framework. For information on log4j, see https://logging.apache.org/log4j/2.x .

The following structured information is prepended to the log before the message, delimited by pipes:

- **Logging level:** Severity level of the message
- **Thread ID:** Numeric ID of the thread
- **User ID:** Unique identifier of the user's logon ID
- **User session ID:** Unique identifier per user object created on the SAP Mobile Platform server
- **Back end session ID:** Unique identifier for the RF connection to the ERP back end
- **Stack information (class / file / line):** Class file and line number that generated the log message
- **Message:** Actual log message

Removal of Old Logger Instantiations

The previous logger class `com.syclo.sap.logger` is deprecated and any custom code using this logger will be logged using the log4j2 classes going forward.

The log methods from the user / server classes are also deprecated and their output now default to the `DEBUG` level.

Options in the javaBE.ini [LOGGING] Section

The following are configuration options in the `javaBE.ini` file, in the `[LOGGING]` section:

- **LOG_DIR:** (string) Specifies base logging directory in which to put the logs. Defaults to `<pwd>/log/agentry` directory if not supplied.
- **CONFIG:** Option to override configuration with a custom configuration, using a configuration file compatible with log4j2's configuration. For more information see http://logging.apache.org/log4j/2.x/manual/configuration.html . By default, this is unused (empty). When specified, all other logging options are ignored.

- **STACKDATA:** (Boolean) If set to `true`, will capture a stack trace when logging that is used to determine the method and line number of the log entry and will also be added to the log output's structured information. Defaults to `false`.
- **CONSOLE:** (Boolean) If set to `true`, will stream all logs out to console as well as the log file. Defaults to `true`.

Logging Calls Use New Objects

A logger object is available to most classes, and logging levels of messages use the following levels:

- **FATAL:** Something that would halt the entire application. For example, errors on start up, or errors on an initial service user connection to retrieve configuration from the back end.
- **ERROR:** Exceptions or BAPI errors
- **WARN:** Warnings from a BAPI
- **INFO:** High level information, for example,
  - User logged in / out, re-logged in
  - Complex table processing, number of rows
  - Data table processing, number of rows
  - Fetch processing, number of objects returned
- **DEBUG:** General debug information, for example,
  - Stack traces from exceptions
  - Basic JCo information when running a BAPI call
- **TRACE:** Comprised of highest-detail information, such as,
  - Individual IDs for records in fetches and complex tables
  - Extra JCo information when running a BAPI call
  - Previous begin (info level) messages from the log that let you know you have entered or exited the function

8.1 Custom Logging Configuration

```
Console Log Layout
%-5level|%i|%X{USERNAME}|%X{INSTANCEID}|%X{SESSIONID}|%X{METHODLABEL}|%msg%n
File Log Layout
%d{yyyy.MM.dd HH:mm:ss.SSS}|%-5level|%i|%X{USERNAME}|%X{INSTANCEID}|
%X{SESSIONID}|%X{METHODLABEL}|%msg%n
Console Log Layout with Location Data
%-5level|%i|%X{USERNAME}|%X{INSTANCEID}|%X{SESSIONID}|%l|%X{METHODLABEL}|%msg%n
File Log Layout with Location Data
%d{yyyy.MM.dd HH:mm:ss.SSS}|%-5level|%i|%X{USERNAME}|%X{INSTANCEID}|
%X{SESSIONID}|%l|%X{METHODLABEL}|%msg%n
```

9 Localization and Translation

SAP mobile applications provide full support for localization of the mobile application without directly modifying the definitions of the application project, including 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: `ApplicationTextBase.ini`. The publish wizard provides a checkbox 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 server's properties that can impact the behavior of the localization functionality. These items can be configured using the SAP Control Center. These are the following options 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 that 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 references 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 Agentry 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 client regardless of the application deployed. Values are displayed in the menus and the 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.
- **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 override values exist in one file, it is not necessary to create 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 issues 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 localization 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 localization 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 localization 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 localization method displays the same override values on all clients, regardless of the devices' locale settings; the multi-language localization method uses the clients' locale settings to determine the required override values.
- Use of the multi-language localization 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 localization method also requires the configuration of client devices with the proper locale settings. The single language localization method does not make use of locale settings and therefore does not require this information.

Regardless of which localization 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 localization method.

9.1 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 allows for different localization override values to be used for different groups of users without the need for multiple 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 locale of a supported language for the application. The language name indicates the language used (i.e., English, French, Spanish). The locale is the sub-language, dialect, or region for the language (i.e., 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 at Language Identifier Constants and Strings .

This Web 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 locale 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 localization method and multi-language localization method 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.

9.1.1 Localizing an Application with the Multi-Language Localization Method

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 Language Identifier Constants and Strings .
- 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 server. The default location is the `localizations` sub-folder in the 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 the `ClientStringFile`, `ApplicationStringsFile`, and the `ApplicationGlobalsFile`.

- Before making any changes to the `Agentry.ini` file, always make a backup copy of the current file, so you can revert any changes if necessary.

Context

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 localization method if needed. However, that process is not discussed here, with the focus on the multi-language method of localization.

Procedure

1. Open the `Agentry.ini` configuration file in your preferred 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, you can add it.

   Here is an example of adding a `localizations` options list, 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, 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 that 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
   Globals.es-ES.ini
   ApplicationText.es-MX.ini
   ClientText.es-MX.ini
   Globals.es-MX.ini
   ApplicationText.es.ini
   ClientText.es.ini
   Globals.es.ini
   ```
6. Once this procedure is complete, restart the server.

   The new override values are read in during the restart.

Results

Once this procedure is complete, the server uses the locale information provided by the clients to determine the proper override values, if any, to use. Those 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 clients by the server.

9.2 Localization: Single Language Support

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

These localization override files are then placed in the installation folder of the 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=Globals.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.

9.2.1 Localizing an Application with the 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 Agentry Administration client. If changes are made to the `Agentry.ini` file directly, make a backup copy prior to modifying the file.

Context

Use this procedure to localize an application using the single language localization method (ex: 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 localization method if needed. However, that 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.

Procedure

1. Copy the translated or localized versions of the localization override files to the folder for the server instance for your application, for example, `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.

Results

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

9.3 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.ini` 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 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, like a built-in screen, or definition type, or 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.

9.4 Override File Format: ApplicationText.ini

Shown here are examples 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 an 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.

9.5 Override File Format: Globals.ini

Shown below are examples of the contents of the `Globals.ini` and `GlobalsBase.ini` files, followed by descriptions 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]`. The 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 Agentry 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.

10 Installation Troubleshooting

10.1 Verifying Version Information

Client Version

Version information for the Windows build of the SAP Work Manager client is found in the `about.txt` file. The `about.txt` file is located in the directory where the client is installed.

For iOS and Android clients, the version can be obtained by viewing the *About* screen, displayed by the client.

10.2 SAP Work Manager Server to the Agentry Component Connection Issues

If the SAP Business Technology Platform Mobile Services did not install correctly when the SAP Work Manager, the application could crash, or not start at all.

If the Agentry component was installed incorrectly, no error logs, or few error logs, are generated as well. To resolve the issue, adjust the classpath to remove the concrete log4j logging libraries. Then, add the `log4j-to-slf4j.jar` bridging jar file to your classpath as shown in the following example:

```
classPath=./Java;./Java/workmanager-6.4.1.0.jar;./Java/common-20161003.jar;./
Java/sapsso.jar;
./sapjco3.jar;./Java/ini4j.jar;./Java/log4j-api.jar;./Java/log4j-to-slf4j.jar;
```

10.3 SAP Work Manager Server to SAP ERP Connection Issues

If you used the connectivity test after installing and configuring the SAP Mobile Platform, connectivity problems can manifest as error messages when you start the server, or as connect test failures.

If errors occur, first check the events log. The event log is the primary troubleshooting tool for connectivity issues. The events log lists either the SAP ERP-specific error code or SAP Work Manager-specific error message.

After installing the SAP Mobile Platform and configuring the connection with your back-end system, if the connection test fails, or you received an error when starting the SAP Mobile Platform, perform the following checks:

1. Verify that the Java classpath in `Agentry.ini` is correct.
2. Verify that the Windows environment PATH variable is correct.
3. Try to connect to the back-end system from the SAP Mobile Platform using the SAP GUI. Verify the user name and password used to connect are correct.
4. Ensure that the client version and the back-end system version numbers are correct.

10.4 SAP Work Manager Server to SAP Work Manager Client Connection Issues

A client connectivity problem will manifest as a *not all data transmitted, logging request failed* message on the client.

If this occurs, check the events log. The events log is the primary troubleshooting tool for connectivity issues. The events log lists all SAP ERP-specific and SAP Work Manager-specific error messages.

Login Request Failed

This error message results when the client device is not communicating with the SAP Mobile Platform. To troubleshoot the above error message:

1. Make sure that the SAP Mobile Platform is running.
2. If the server is not started, you will see the logging in message for a minute and eventually get this error.
3. Check the events log to make sure that the SAP Mobile Platform has not lost its SAP ERP connection. Check for the following network issues:

- Time-outs and TCP/IP issues: Troubleshoot network
- Wireless network issues: Troubleshoot wide area network
- Firewalls: Ensure that configured ports are open for the IP addresses of the clients

Incorrect User ID or Password

This error is seen on the mobile device when an incorrect user ID or password is used. View the error in the `events.log` file located in the install directory of the SAP Mobile Platform.

To fix the problem, have the technician use the correct password.

Failure to Connect to the Server (3)

This error is received when the SAP Work Manager client fails to connect to the SAP Mobile Platform or an established connection is lost. Because a connection to the server was not made or was lost, there will not be an entry in the `events.log` file.

A common cause of this error is that a client's network connection is not established or is severed. If this error is received during transmit, note that some data may have already been updated to the back end system.

To fix the problem, re-establish the client device's network connection and perform another transmit. Any data not sent during the previous transmit will be sent during the next successful transmit, and will be listed in the transmit dialog of the SAP Work Manager client.

Communication Error (14)

This error is received when the SAP Work Manager client cannot communicate with the SAP Mobile Platform. Because there is a problem communicating with the server, there is not an entry in the `events.log` file. Common causes of this error are:

- The SAP Mobile Platform service is not started
- The network connection is not working properly
- The device has production logic on it and is now trying to connect to the SAP Mobile Platform. This is shown in the `events.log` file.

To fix the problem:

- Start the SAP Mobile Platform service.
- Disconnect, then re-establish a network connection.

10.5 Message Codes

All communication between the SAP Mobile Platform and the SAP Work Manager client are logged to a log file on the server named `messages.log`. A transmission from the SAP Work Manager client to the SAP Mobile Platform is made up of many individual messages. Each message goes through several states as it is processed by the SAP Mobile Platform. A single line is written to the `messages.log` for each state of each message per user.

The log file is located in the directory where the SAP Mobile Platform was installed. If the server was installed in the default directory, it can be found here for the SAP Mobile Platform:

```
C:\SAP Work Manager\ServerProd\Logs
```

The messages log is stored in the following location for the SAP Mobile Platform:

```
C:\SAP Work Manager\ServerDev\Logs
```

Message Codes

Code Number	Code Message / Description
1	SystemInfoRequestRC
2	LogoutRequestRC: Client logout from server
3	Login Request: Client login from server
5	LogoutNoticeRC: The server decided to log the user out (the user timed out, the server or the app is shutting down, an administrator disconnected the user, etc.)
6	ResetRequestRC
7	ChangePasswordRequestRC: Client attempt to change password
8	PublicKeyRequestRC
11	DecryptKeyRequestRC
12	UploadE2ETraceRequestRC
13	PasswordValidationAuditRecordNoticeRC
200	Object Transaction: Client send of a single transaction to the server
201	Object Fetch: Client request for a fetch to be run by the server and the result objects to be sent to the client
202	Client System Info Notice: Client-specific information sent to server such as client hardware, OS, screen size, etc.
203	Object Refresh: Client request for server to send updated copy of an object
204	Object Definition Request: Client request for an Agentry object definition
205	Fetch Definition Request: Client request for an Agentry fetch definition
206	Transaction Definition Request: Client request for an Agentry transaction definition
207	Screen Set Definition Request: Client request for an Agentry screen set definition
208	Action Definition Request: Client request for an Agentry action definition
209	Rule Definition Request: Client request for an Agentry rule definition
210	Report Definition Request: Client request for an Agentry report definition
211	Object Push: Server send of objects and/or messages sent to the client
212	Enable Push: Sent by client to enable push for this user on the server
213	Style Definition Request: Client request for an Agentry style definition
622	Complex Table Request: Client request for all Agentry complex table updates
623	Data Table Update Request: Client request for all Agentry data table updates. The formatting of <TEXT> following a <p> is not supported. Wrap the <TEXT> that follows in a <p>.

Error Codes

Code Number	Code Message / Description
0	NoFailure: No failure, not worth mentioning
1	UnknownFailure: A catch-all failure error code
2	NotLoggedIn: Client thought it was logged in, but the server did not think it was logged in
3	GrapevineCreateFailed: An attempt to create a grapevine failed
4	ChangingMediums
5	MediumConnectFailed
6	MediumNotAvailable
7	AttemptCancelled
8	MessageCancelled
9	TimedOut
10	NoEscalateGroupName
11	GrapevineDisconnectedUnexpectedly
12	ReceiverCoudNotDecode: The message receiver could not understand the message
13	ReceiverException
14	CommunicationError
<no code number>	AuthenticationError
<no code number>	PasswordChangeFailed
<no code number>	PasswordChangeFailedAccountBlocked

Push Message States

State	Name	Description
O	Outgoing	Message in process of being sent to client
T	Trying	Attempting to connect to client

State	Name	Description
L	Linked	Successfully connected to client
W	Waiting	Failed attempt to connect to client, will retry
R	Received Response	Client response received by server
S	Sent response	Server sent information and/or acknowledgement to the client
C	Complete	Message complete
X	Cancelled	Message cancelled by server
F	Failed	Message failed, will not retry. The formatting of <TEXT> following a <p> is not supported. Wrap the <TEXT> that follows in a <p>.

Message States

State	Name	Description
I	Incoming	Message in process of being received from client
Q	Queued	Message has been decoded, user has been identified, and the message is placed in one of the server's work queues
S	Sent response	Server sent information and/or acknowledgement to the client
R	Received Response	Client response received by server
C	Complete	Message complete. The formatting of <TEXT> following a <p> is not supported. Wrap the <TEXT> that follows in a <p>.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP