PUBLIC
SAP Asset Manager
Document Version: 4.0 1911 – 2019-12-04

SAP Asset Manager Component 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 https://help.sap.com/viewer/product/SAP_ASSET_MANAGER/p/en-US.

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

Document Version	Date	Description of Changes
1911	NOV 2019	Original release

1 Overview of Additional Components for SAP Asset Manager

In addition to the core functionality of the SAP application, there are also additional, optional components that can be installed. There are different deployment executable files for SAP Asset Manager:

- SAP Asset Manager base
- SAP Asset Manager with Field Operations Worker
- SAP Asset Manager with Meter Management
- SAP Asset Manager with Crew Management
- SAP Asset Manager with Customer Service
- SAP Asset Manager with Asset Central

The file you use for your installation depends on your business needs. The sections in this guide walk you through each of the component installation scenarios.

1.1 Common Procedure: Enabling a Component in SAP Asset Manager

Extend the SAP Asset Manager application with the functionality of your selected component.

Prerequisites

- Verify that your system is set up to build the SAP Asset Manager application with your selected component by running the Mobile Development Kit Dependencies Installer tool. This tool detects all the components to install or update, allowing you to update or install them instantly.
  For more information and instructions on how to obtain the `Dependencies Installer.app` tool, see the Building Your MDK Client SDK procedure.
- Complete the Creating an Application in SAP Cloud Platform mobile services procedure.

Context

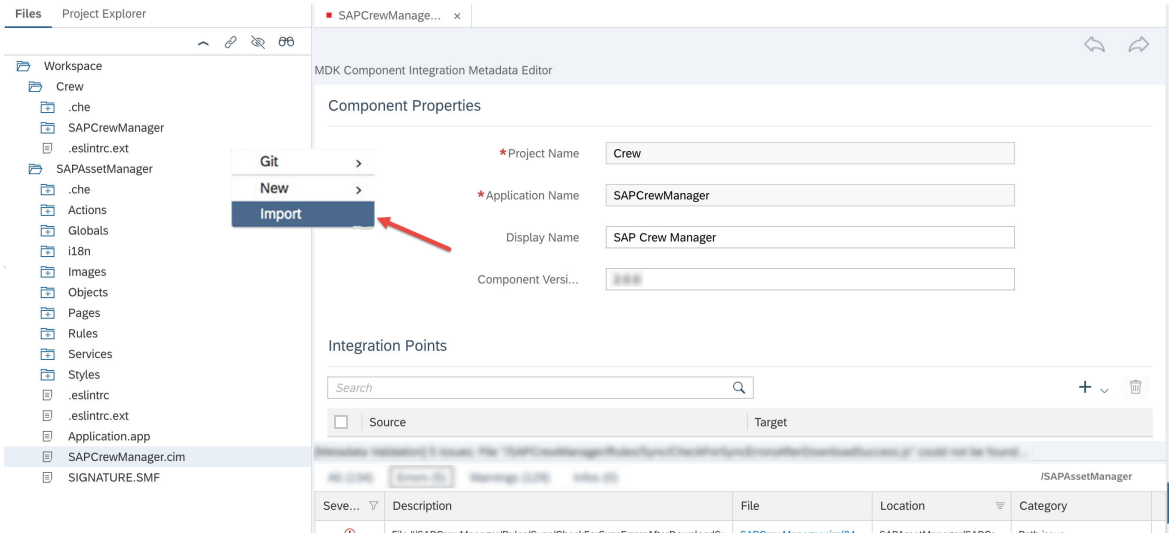
You can use the following procedure for any of the components available with the SAP Asset Manager application: Meter Management, Field Operations Worker, Crew Management, Customer Service, or Asset Central.

The following procedure uses the Crew Management component to walk you through the installation process. To install other components, replace the Crew Management file names you see in the procedure steps and screenshots with the name of the component you are installing.

> **Note**
>
> To build a branded component application, see the procedure on how to build a branded application in the specific component chapter. For example, to build a branded Crew Management component, see the Branding the Crew Management Component for SAP Asset Manager [page 38] procedure in the SAP Asset Manager with Crew Management [page 36] chapter.

Procedure

1. Open the SAP Web IDE. From the Mobile Development Kit perspective (), right-click on *Workspace* and select *Import* from the menu.

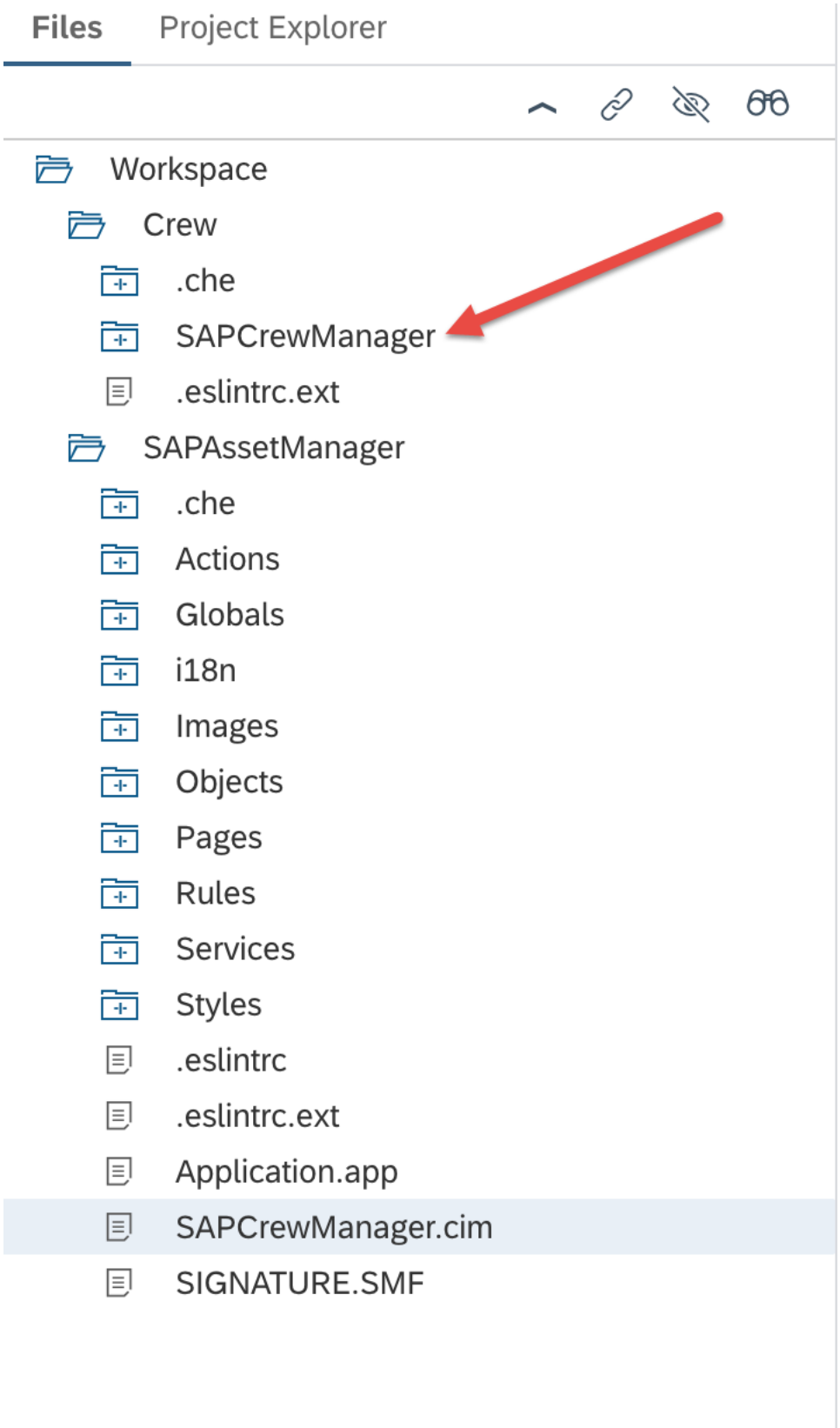


2. From the *Import* window, use the *Browse* button to navigate to your ZIP file of the component metadata on your file system.
3. Select the component ZIP file to import. Change the component file path in the `<Import To>` field based on the component you are importing to the following:
   - `/Crew/SAPCrewManager`
   - `/Meter/SAPMeterManager`
   - `/FOW-Component/FieldOperationsWorker`
   - `/CustomerService/SAPCustomerService`
   - `/AssetCentral/SAPAssetCentral`

   > **Note**
   >
   > If you are creating your own CIM file, your project name must match the folder name and the application name must match the subfolder name.

4. After selecting the ZIP file and changing the file path name in the `<Import To>` field, click *OK* to continue.

   After importing the ZIP file, you see the component definition folder created in the *Workspace* folder.


Files
Project Explorer
Workspace
Crew
.che
SAPCrewManager
.eslintrc.ext
SAPAssetManager
.che
Actions
Globals
i18n
Images
Objects
Pages
Rules
Services
Styles
.eslintrc
.eslintrc.ext
Application.app
SAPCrewManager.cim
SIGNATURE.SMF

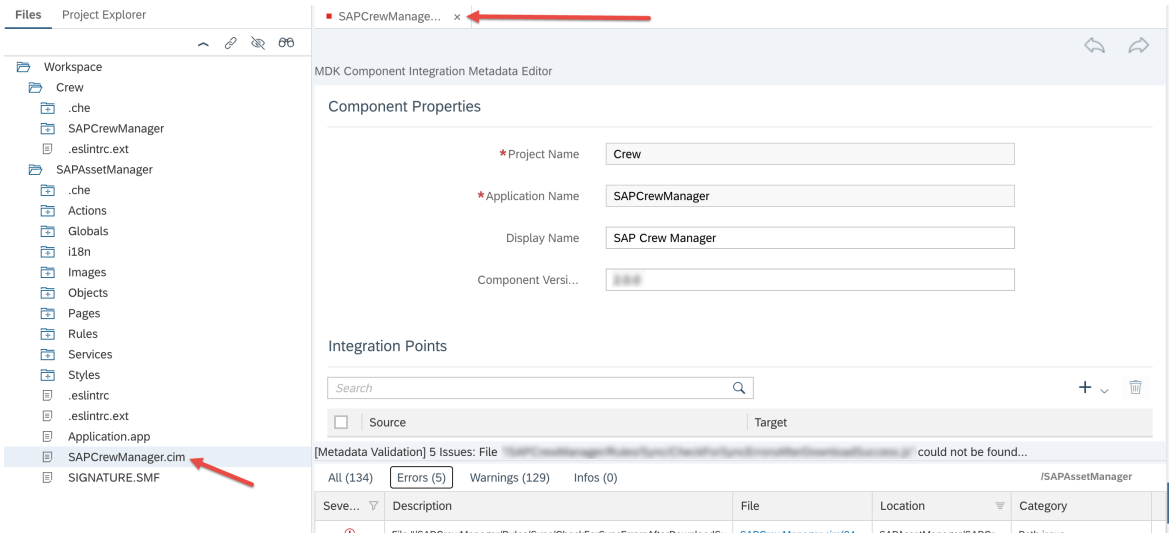
5. Import the Component Integration Mapping (CIM) file for your component into the *SAPAssetManager* project folder, so that you can incorporate the component into the SAP Asset Manager application. You can find the CIM file in the following location:

   `Branding SDK/<ComponentName>/SAM.mdkproject/metadata/sam.definition/`

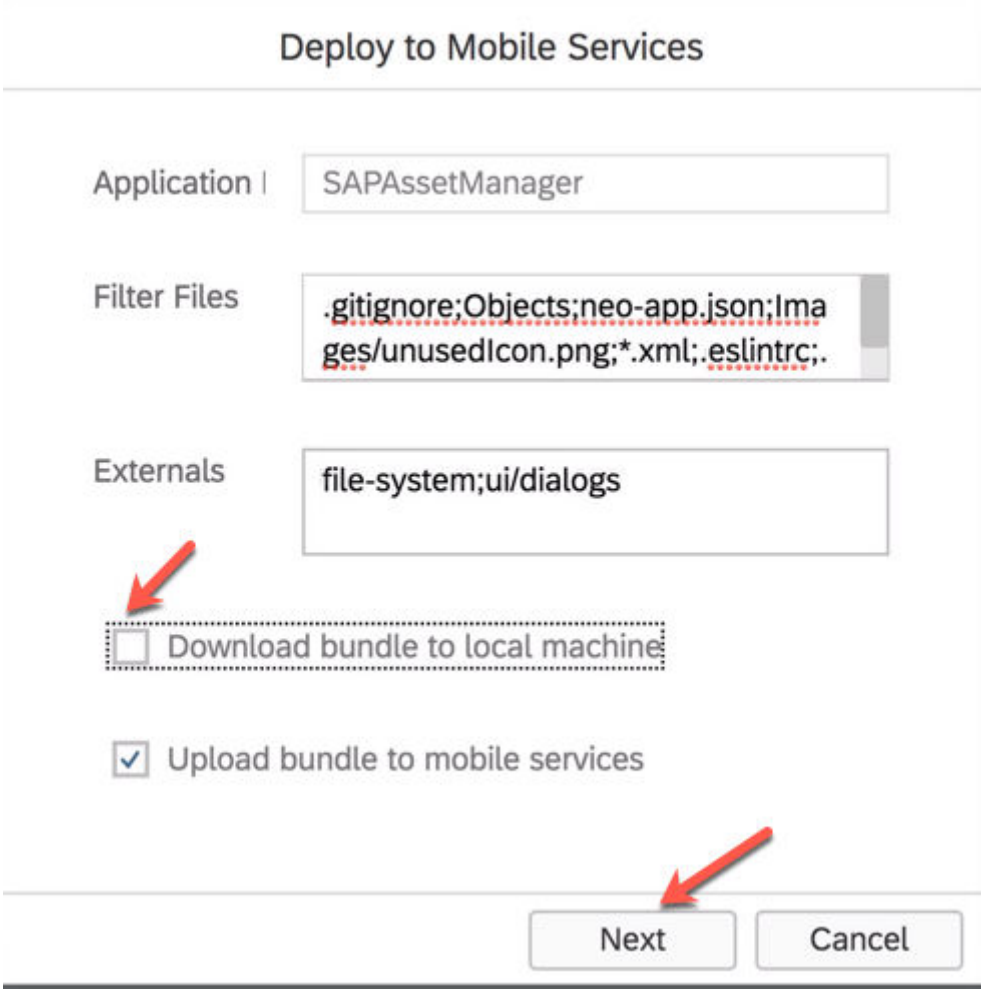
   > **i Note**
   >
   > Your *Project Name* of the CIM file must match with the folder name, or you will receive an error.

   a. In your *Workspace*, right-click on the *SAPAssetManager* folder and select *Import*.
   b. Using the *Browse* button, select the CIM file for your component where you previously unzipped it.

   After you import the `SAP<Component>Manager.cim` file to your *SAPAssetManager* folder, you will see the details of the CIM file in the workspace. If a metadata error message appears, you can ignore the message.



6. Redeploy the *SAPAssetManager* project to SAP Cloud Platform Mobile Services.
   a. Right-click on the *SAPAssetManager* folder and select *MDK Deploy and Activate*.
   b. In the Deploy to Mobile Services window, ensure that the following information appears in the `<Filter Files>` and `<Externals>` fields as seen in the example screenshot. Uncheck the `<Download bundle to local machine>` checkbox. Click *Next* when finished.



c. In the Deploy to Mobile Services - Destination window specify the `<Destination Name>` and `<Application ID>` to where the application is deployed in SAP Cloud Platform Mobile Services. When finished, click *Next*.
d. Click *Next* to generate the webpack and to build your project.

Results

Observe as the *MDKWebpackFactory* project is either updated or created. The console window then displays the deployment detail. When the project creation is complete, the console shows the successful deployment of the application as well as the application revision number.

Next Steps

After your successful deployment, the next time the *Check for Updates* message displays on the SAP Asset Manager client, click *OK* to switch to the updated definitions.

Related Information

https://launchpad.support.sap.com/#/softwarecenter/template/products/%20_APP=00200682500000001943&_EVENT=DISPHIER&HEADER=Y&FUNCTIONBAR=N&EVENT=TREE&NE=NAVIGATE&ENR=73555000100200009531&V=MAINT&TA=ACTUAL&PAGE=SEARCH/ASSET%20MGR%20BRAND%20SDK%20IOS%203.0

1.2 Allowing Custom URI Schemes

Prerequisites

You have built and branded the SAP Asset Manager application. For more information, see the following topics:

-
-

Context

> **i** Note
>
> The following procedure is applicable for iOS installations only. You do not need to perform this procedure if you are installing SAP Asset Manager on Android.

By default, iOS allows third-party apps to specify a limited set of URI schemes:

- http:
- https:
- mailto:
- tel:
- sms:
- facetime:

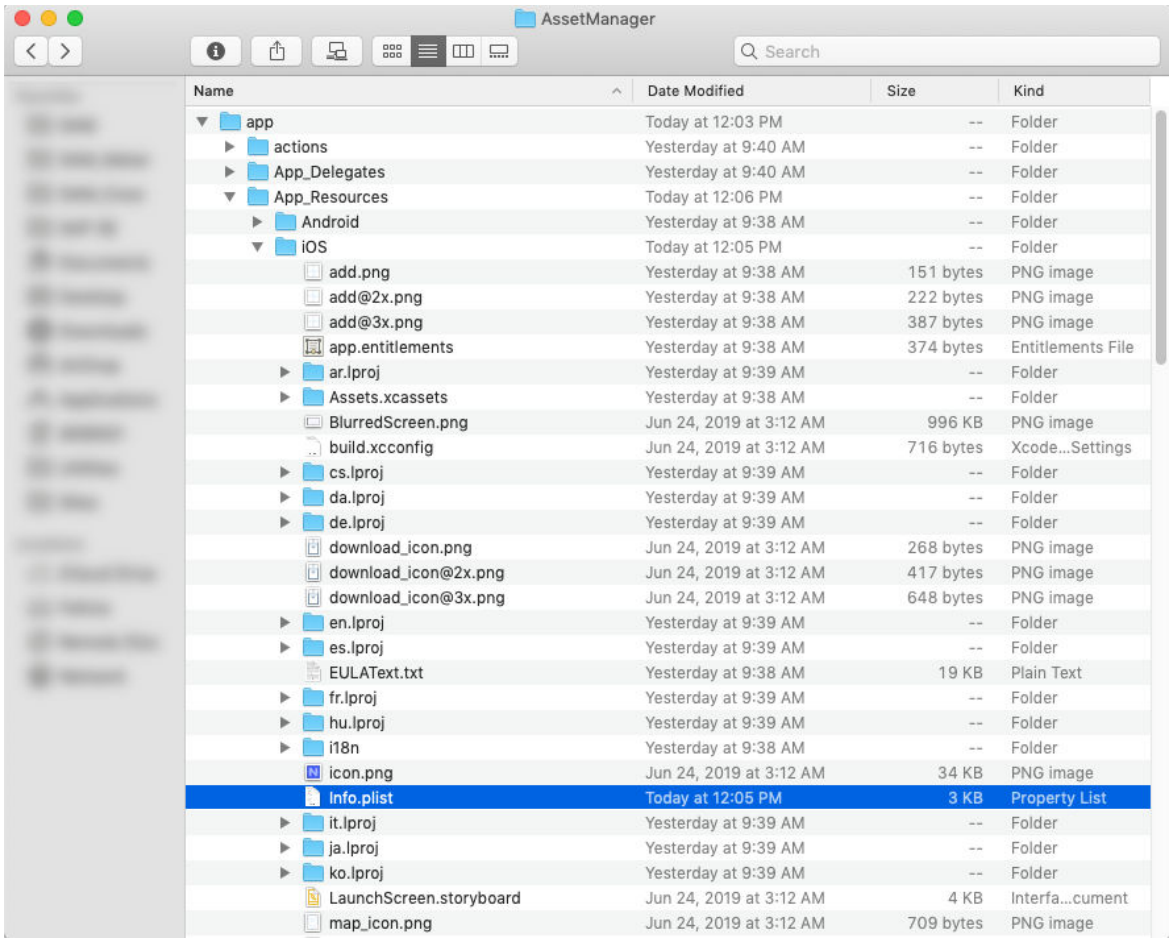
Other applications, such as Microsoft Edge, can support custom URI schemes. For example, if the application is installed, `microsoft-edge-https://www.google.com`, opens up Google in Microsoft Edge for iOS. However, it will only work correctly from a non-third-party iOS application such as Safari.

For additional information on iOS URI schemes, see the Launch Services Keys documentation from Apple.

To allow SAP Asset Manager to open custom URIs, use the following procedure.

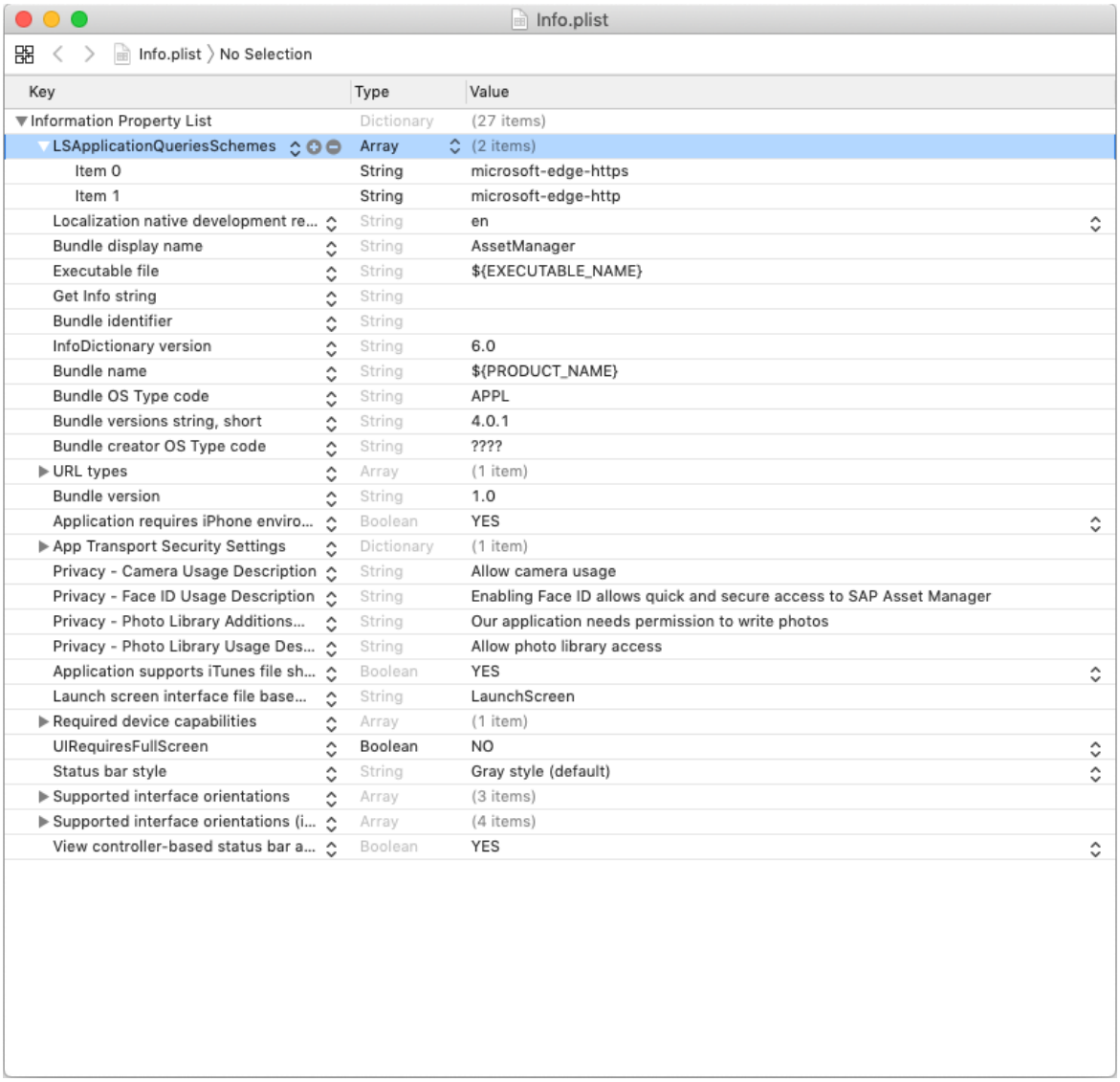
Procedure

1. Locate the `Info.plist` file, located at `${ASSET_MANAGER_ROOT}/app/App_Resources/iOS`, after building the SAP Asset Manager client.



2. Open the `Info.plist` in Xcode, and add a new key named `LSApplicationQueriesSchemes` of type *Array*.
3. Add an entry for each custom URI scheme. Don't include any trailing colons or slashes.

   In the following example, the custom schemes `microsoft-edge-https` and `microsoft-edge-http` are added. If Microsoft Edge is installed on the mobile device, these custom schemes allow the client to open HTTP and HTTPS URLs in Microsoft Edge.



4. Rebuild and rerun the SAP Asset Manager client.

Results

Any custom URL attachments associated with your newly added schemes now open as expected.

1.3 Enabling and Disabling Component Features Per User Through SAP Authorization

Using parameter framework configuration, configure parameters to enable or disable various features per the authorization of the user in the back end.

Each mobile user is connected to a back end SAP user. The back-end SAP user can be assigned one or more roles. These roles grant their holders authorizations within the back end system. Through parameter configuration, SAP provides a standard rule handler that performs a TCode authorization. SAP also provides new globals that can turn on and off new features.

If a parameter is enabled to use a rule instead of a global, and the user role has an authorization to run a specific transaction code, then that specific feature is enabled for that SAP user. If the user has the authorization' for a specific transaction code, then the specific feature is disabled for that mobile user. Therefore, depending on the authorization of the SAP user, the feature now either works or doesn't work, displays, or doesn't display (depending on the feature function), rather than is turned on or off for all users.

For detailed information on how to configure features, see Enabling and Disabling Features Per User Through SAP Authorization in the *SAP Asset Manager Configuration* guide.

Features Available Through SAP Authorization

The following features are available for you to enable or disable starting with the SAP Asset Manager 1911 release. Use the following subsection to learn how to use the ConfigPanel to enable or disable a feature based on the authorization of the user.

Component	Functionality	Category	TCODE	Back-End Parameter	Comments
CUSTOMER SERVICE	Service notification create	Notifications	IW51	Enable.SNO.Create	
CUSTOMER SERVICE	Service notification edit	Notifications	IW52	Enable.SNO.Edit	Except local
ASSET CENTRAL	Add checklist	Checklist	N/A	Enable.CL.Create	See the *Generic Authorization Check* section in Enabling and Disabling Features Per User Through SAP Authorization
ASSET CENTRAL	Fill checklist	Checklist	N/A	Enable.CL.Edit	See the *Generic Authorization Check* section in Enabling and Disabling Features Per User Through SAP Authorization
CREW	Manage crew	Crew	N/A	Enable.Crew.Manage	See the *Generic Authorization Check* section in Enabling and Disabling Features Per User Through SAP Authorization

1.4 Integrating Multiple Components

To integrate multiple add-on components to the base SAP Asset Manager application, a new 'integrator' component is created and used.

Context

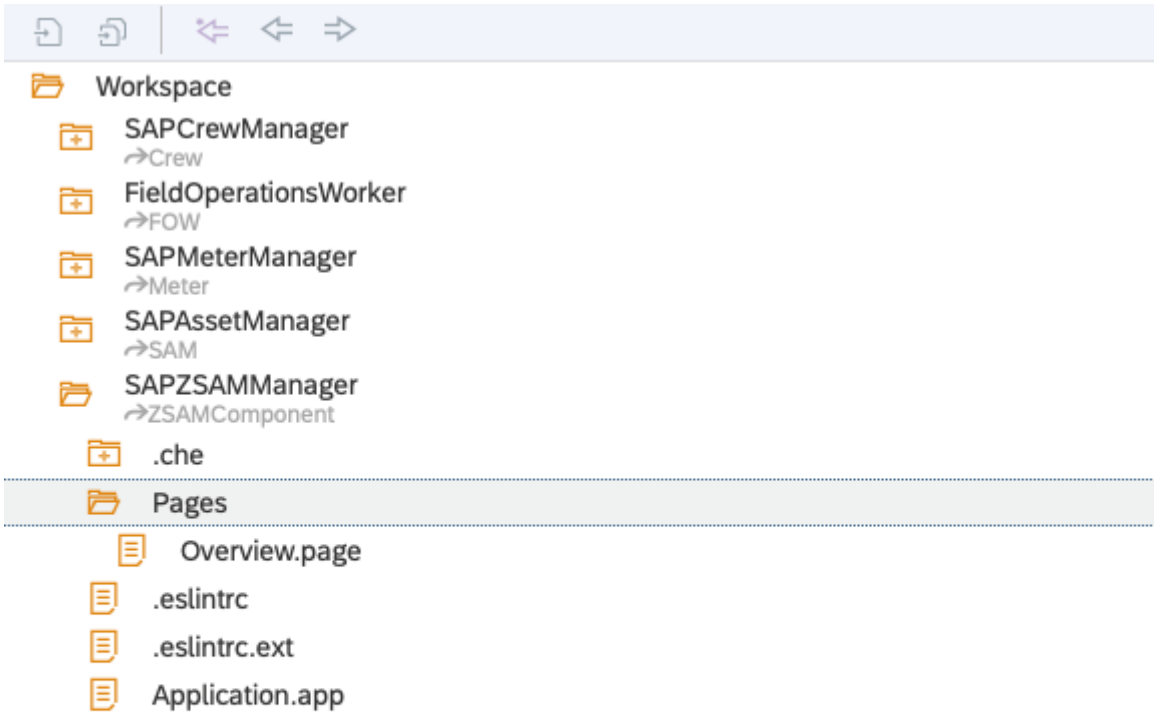
The integrator component includes only files common to two or more components. These common files are the only files that require merging. When you shift these merged common files to the integrator component, you leave the original components as-is or untouched. Using this method, it is easier to upgrade any add-on component when a new version is released.

For example, consider creating the SAP Asset Manager application with the Crew Management and Field Operations Worker components. Both these components have an `Overview.page` as a common file, as well as other common files. Therefore, you create a new *MyAggregate* component. You then add an `Overview.page` to the *MyAggregate* component, where the `Overview.page` is a merged page from Crew Management and Field Operations Worker.

The following procedure details how to merge the Field Operations Worker, Crew Management, and Meter Management components, along with custom changes. Use the procedure as an example. Your components and custom changes may vary.

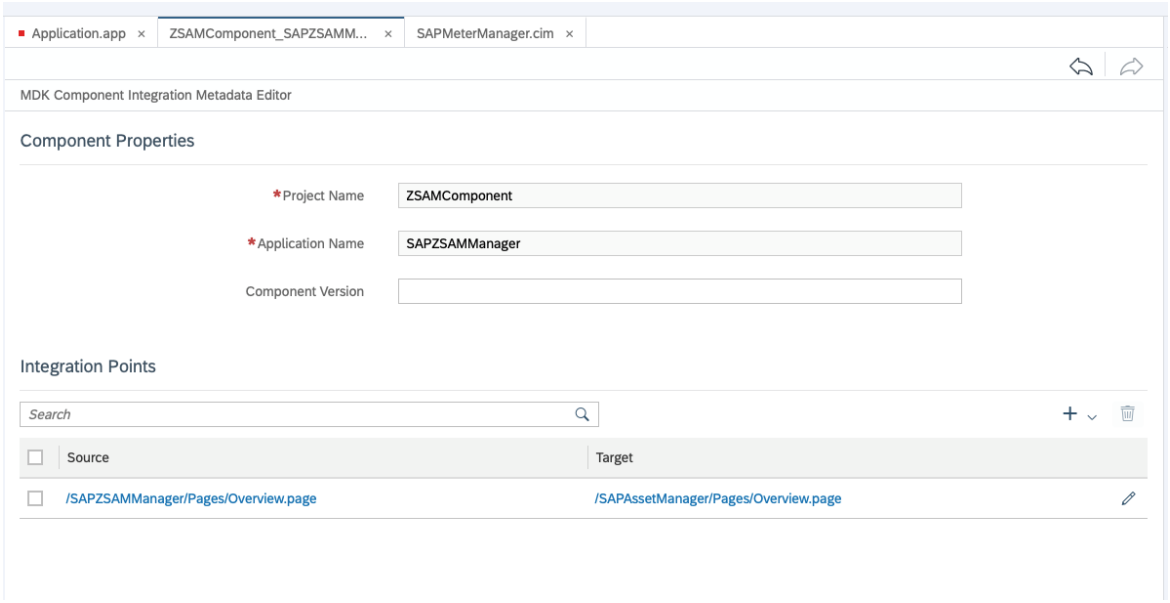
Procedure

1. Create a *ZSAMComponent* with an `Overview.page`.

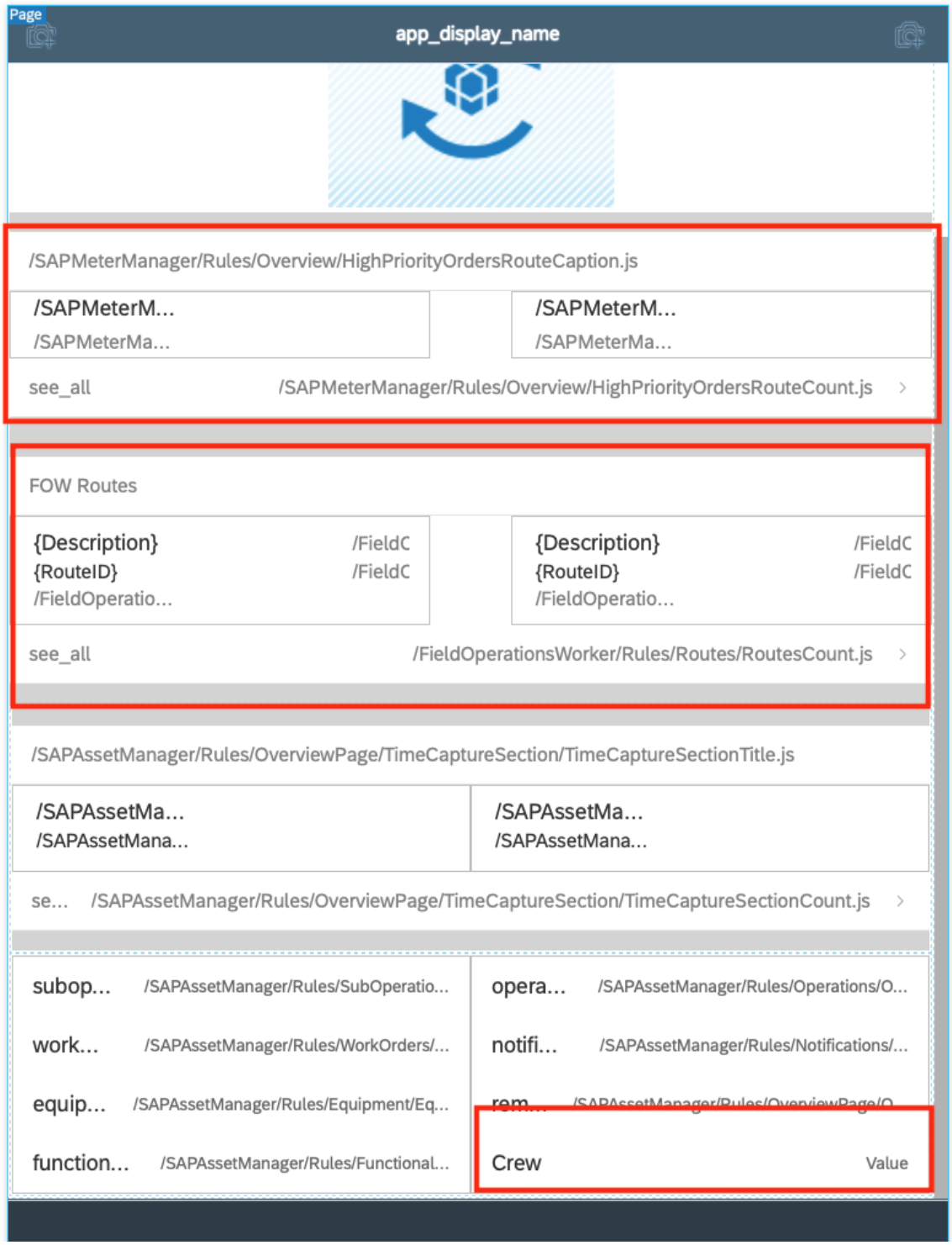


2. Create another integrator component called `ZSAMConponent`. Place the `.cim` files of the components you are installing to the base SAP Asset Manager application into `ZSAMComponent`, as shown in the following screenshot.

> **i Note**
>
> Multiple `.cim` files are executed in alphabetical order. In this case, the `ZSAMComponent` is the last one executed and will override all other changes made by the other `.cim` files.


Application.app
ZSAMComponent_SAPZSAMM...
SAPMeterManager.cim
MDK Component Integration Metadata Editor
Component Properties
*Project Name
ZSAMComponent
*Application Name
SAPZSAMManager
Component Version
Integration Points
Search
Source
Target
/SAPZSAMManager/Pages/Overview.page
/SAPAssetManager/Pages/Overview.page

An example of the *Overview* page metadata of the *ZSAMComponent* after the merger of the *Overview* pages of the Field Operations Worker, Crew Management, and Meter Management components:



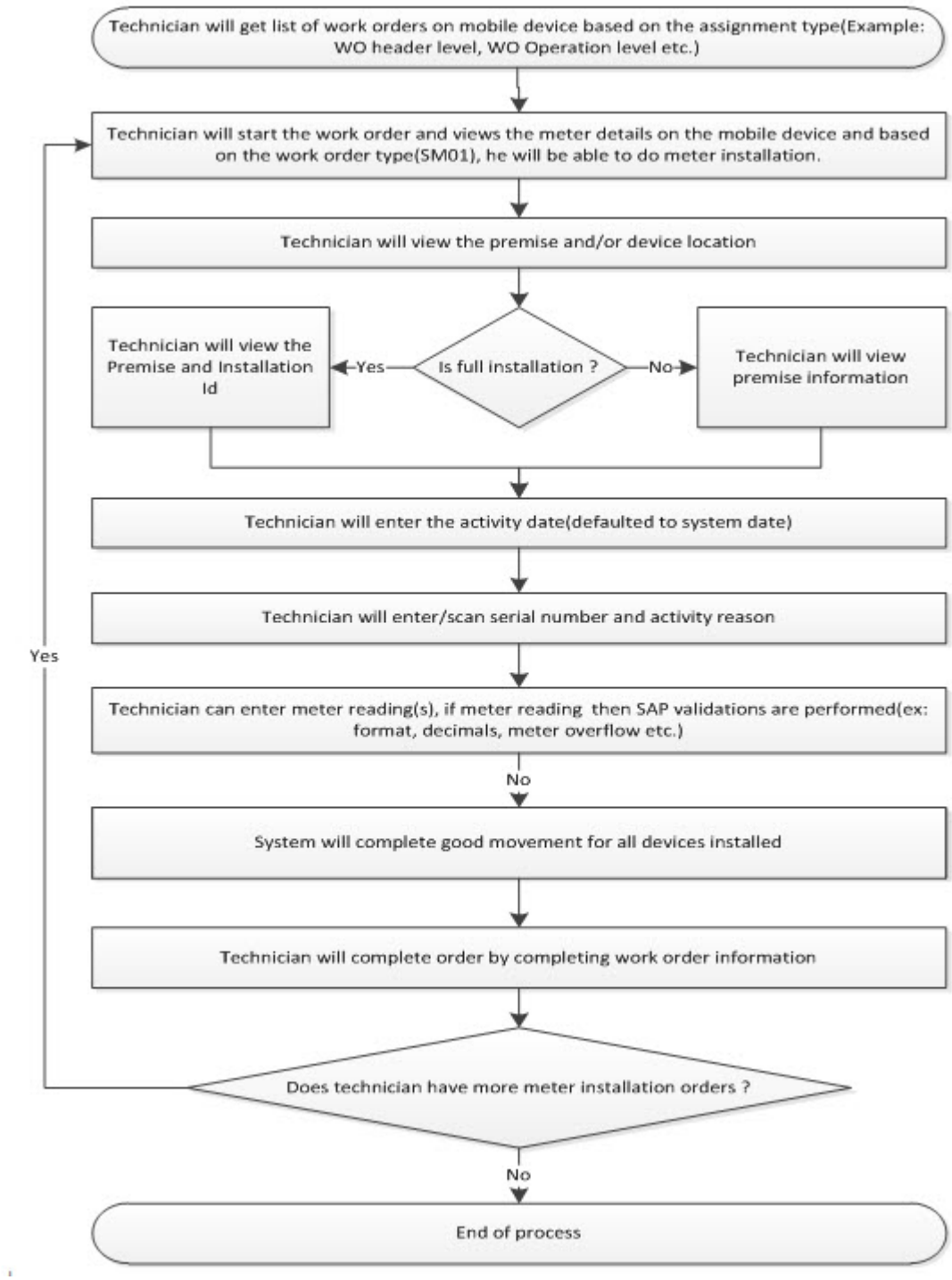
2 SAP Asset Manager with Meter Management

The meter management add-on component for SAP Asset Manager is an optional component that helps utility companies integrate meter data management systems with SAP back-office applications. The solution is designed to connect customer usage data that had been collected from meters with billing, customer care, and other operational and business applications from SAP.

When this add-on component is implemented, it adds the following functionality to the core SAP Asset Manager application:

- Install meters (technical and full installations)
- Remove meters (technical and full removal)
- Replace meters (full replacement)
- Meter repair
- Aperiodic meter readings

The following diagram is a sample of the workflow for meter installation.



2.1 Building the Meter Management Component Overview

Use the following information as a reference when building your application using the procedure Branding the Meter Management Component for SAP Asset Manager [page 20].

Structure of .mdkproject

- **BrandedSettings.json:** Runtime configurations such as security settings, URLs for connecting to the SAP Cloud Platform mobile services, and more
- **MDKProject.json:** Build time configurations such as the application name, version, and bundle ID
- **App_Resources:** Any custom resources used by the application
- **demo:** To make an OData service available in demo mode, include the `.udb` and `.rq.udb` files for that service in this directory
- **extensions:** Include any extensions used by the application in this directory
- **metadata:** Built in metadata for the application

Configuring the MDKProject.json File

The `MDKProject.json` file contains settings that you can only configure before running the `create-client.command`:

- **AppName:** Determines the name of the application project and the app as it appears on an iOS device
- **AppVersion:** The client project application version
- **BaseProject:** The `metadata` subdirectory under the `.mdkproject` structure that contains the main application metadata. The main application metadata is the MDK application, which includes one or more component MDK applications. The component applications are only required if you are adding components to your base application, such as Meter Management or Field Operations Worker.
- **BundleID:** Uniquely identifies the resulting MDK client application on the device. Only one instance of a bundle ID can be installed on a device at a time. If you attempt to install a second application using the same bundle ID, it will overwrite the existing application.
- **Externals:** A list of NPM nodules that should not be included in the application bundle. Use this option for dependencies you expect to be in the environment when the application is built.
  Note that the modules `file-system` and `ui/dialogs` are automatically used as externals as they are already included in the client application.
- **URLScheme:** Allows you to specify a custom URL scheme that opens the client. If the URL includes connection settings such as URL parameters, these settings override the settings used by the client. Defaults to *mdkclient*.

Application Version and Notes on the Settings App

The Mobile Development Kit client tracks several versions, which you can view in the iOS *Settings* menu. These versions are identified as the *application* version, the *definitions* version, and the *frameworks* versions for the frameworks used in the client build.

When generating a client project, you can specify the application version. Specifying the application version allows you to version the client itself, which can be useful if you change extension controls or other branded settings. To specify the application version, specify the `AppVersion` property in the `MDKProject.json` file before running `create.client.command`.

To further customize the entry of your application in the iOS *Settings* menu, you can manually edit `<ProjectDirectory>/app/App_Resources/iOS/Settings.bundle/Root.plist` after the script has completed. You can add new entries, but do not remove existing entries or the application may not function correctly.

For more information, see Implementing an iOS Settings Bundle .

2.2 Branding the Meter Management Component for SAP Asset Manager

Deploy the Meter Management component from the out of the box configuration to set the cloud endpoint authentication URL and the OData service URL. You can also set other configuration values.

Prerequisites

- Verify that your system is set up to build the SAP Asset Manager application with the Meter Management component by running the Mobile Development Kit Dependencies Installer tool. This tool detects all the components to install or update, allowing you to update or install them instantly.
  For more information and instructions on how to obtain the MDK Dependencies installer, see the Building Your MDK Client SDK procedure.
- Complete the Creating an Application in SAP Cloud Platform mobile services procedure.

Context

Use the following procedure to build and brand the SAP Asset Manager application with the Meter Management component.

Procedure

1. Obtain the following SAP Asset Manager 1911 and Mobile Development Kit installation files from the SAP Download Center:
   - `MDK Plug-In SDK 4.0 folder`

   > **i Note**
   >
   > Select the following SAP Asset Manager and Mobile Development Kit versions based on your release:
   > - For SAP Asset Manager release 4.0.1, select Mobile Development Kit 3.1.3.
   > - For SAP Asset Manager release 4.0.2, select Mobile Development Kit 3.2.1.
   > - For SAP Asset Manager release 4.0.3, select Mobile Development Kit 3.2.4.
   >
   > The instructions in this procedure use the `MDK Plug-In SDK 4.0 folder` as an example, and may not be reflective of your installation.

- Metadata 4.0 folder for SAP Asset Manager (`ASSET MANAGER METADATA 1911`)
- Metadata 3.0 folder for Meter Management (`ASSET MGR METER METADATA 4.0`)
- Branding SDK for SAP Asset Manager, Meter Management, Field Operations Worker, Crew Management, Customer Service, and Asset Central (`ASSET MGR BRANDING SDK 1911`)
- Plug-ins: `MDK PLUG-IN SDK 4.0`. The SDK contains the SAP Asset Manager extension controls for the MDK.

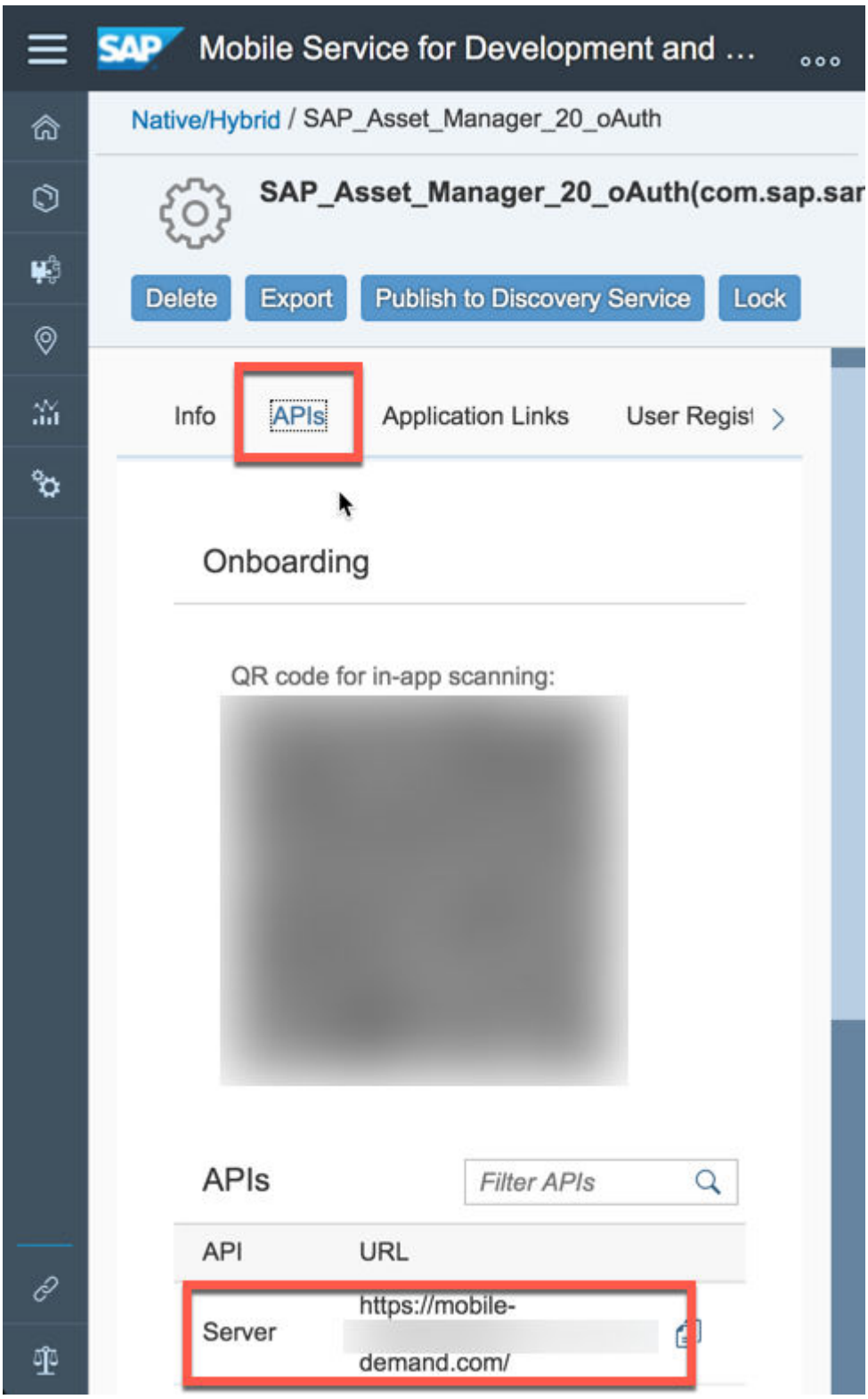
2. Create a folder to contain the installation files (`SAPAssetManager4.0Meter`) on your Mac.
3. Extract the SAP Asset Manager branding SDK:
   a. Unzip `ASSET MGR BRANDING SDK 1911`.
   b. Copy the `SAPAssetManagerMeter/SAM.mdkproject` folder to the `SAPAssetManager4.0Meter` folder.
   c. Set up the SAP Asset Manager Mobile Development Kit project folders:
      1. In the `SAPAssetManager4.0Meter/SAM.mdkproject/metadata` folder, create a folder named `Meter`.
      2. In the `SAPAssetManager4.0Meter/SAM.mdkproject` folder, create a folder named `extensions`.
4. Extract the SAP Asset Manager metadata:
   a. Unzip the `ASSET MANAGER METADATA 1911` file.
   b. Copy all contents of the ZIP file to `SAPAssetManager4.0Meter/SAM.mdkproject/metadata/sam.definitions`.
   c. Delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0Meter/SAM.mdkproject/metadata/sam.definitions` folder, if present.
5. Extract the Meter Management SAP Asset Manager metadata:
   a. Unzip the `ASSET MGR METER METADATA 4.0` file.
   b. Copy all contents of the ZIP file to `SAPAssetManager4.0Meter/SAM.mdkproject/metadata/Meter`.
   c. If present, delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0Meter/SAM.mdkproject/metadata/Meter` folder.
6. Extract the MDK plug-in SDK in one of the following ways, based on whether you're building either an iOS or an Android client:

iOS Client	Android Client
1. **Unzip the iOS subfolder found in the `MDK PLUG-IN SDK 4.0` zip file.** 2. **Choose your architecture from one of the following folders:** ○ **Release-iphoneos** ○ **Release-iphonesimulator** ○ **Release-fat (contains both the iphoneos and the iphonesimulator architectures)** 3. **Copy the following folders from your selected architecture folder to the**	1. Unzip the Android subfolder found in the `MDK PLUG-IN SDK 4.0` zip file. 2. Copy the following folders from your `Universal` folder to the `SAPAssetManager4.0Meter/SAM.mdkproject/extensions` folder: ○ `extension-Analytics` ○ `extension-BarcodeScanner` ○ `extension-FieldDataCapture` ○ `extension-MapFramework` ○ `extension-HierarchyFramework`

iOS Client	Android Client
`SAPAssetManager4.0Meter/SAM.mdkproject/extensions` **folder:** ○ `extension-Analytics` ○ `extension-BarcodeScanner` ○ `extension-FieldDataCapture` ○ `extension-MapFramework` ○ `extension-HierarchyFramework`	

7. Configure the connection to SAP Cloud Platform mobile services:
   a. Retrieve the following information to establish a connection between the SAP Asset Manager application and the SAP Cloud Platform mobile services:
      - **AppId:** Note the ID under the Mobile Development Kit that you created in *Step 6* of the Creating an Application in SAP Cloud Platform mobile services procedure.
      - **ClientID:** Note the oAuth client ID that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

- **SapCloudPlatformEndpoint:** Find the Endpoint setting inside the application list of APIs on the *Mobile Services* under the *Server* API:



> **Note**
>
> By default, the *Server API* has a / at the end of the endpoint URL. Don't add this / into your connection settings.

- **AuthorizationEndpointURL:** Note the oAuth authorization endpoint URL that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **RedirectURL:** Note the callback URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **TokenURL:** Note the token URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

b. Choose your client configuration:

To preconfigure your client to connect to your mobile application, add the information retrieved in the above step to the *ConnectionSettings* block. When adding additional entries, include a comma after the existing *EnableOverrides* entry. An example is included in the table for reference when you're adding entries to your file.

If you want to use SAP Asset Manager against different back-end mobile applications (ex: DEV and QA), leave the *ConnectionSettings* as is, and build an onboarding URL for users using the values found in *Step 7a*. See the example in the table for further information on how to connect a client using either of the methods.

Preconfigured Client	Onboarding URL

To configure the client to connect to a specific application, update the following values in the `BrandedSettings.json` file:
 	
Sample Code
`…`
`"DetailLabelViewText": "$(L,detail_label_view_text)",`
`"ConnectionSettings": {`
`"EnableOverrides": true,`
`"AppId": "<Insert AppID value here>",`
`"ClientId": "<Insert ClientID value here>",`
`"SapCloudPlatformEndpoint": "https://<Insert cloud platform endpoint URL here>",`
`"AuthorizationEndpointUrl": "https://<insert authorization endpoint URL here>",`
`"RedirectUrl": "https://<insert redirect URL here>",`
`"TokenUrl": "https://<insert token URL here>"`
`},`
 	
Using the example as a guide, insert your own connection-specific values where they belong.
 	
Save any changes you make.
	You can generate an onboarding link to overwrite the values discussed in this substep on a device. Use the following format in a URI:
	
	Sample Code
	`samclient://?AppId=<Insert AppID value here>`
	`&ClientId=<Insert ClientID value here>`
	`&SapCloudPlatformEndpoint=<Insert cloud platform endpoint URL here>`
	`&AuthorizationEndpointUrl=<insert authorization endpoint URL here>`
	`&RedirectUrl=<insert redirect URL here>`
	`&TokenUrl=<insert token URL here>`
	`&ServiceTimeZoneAbbreviation=<insert timezone abbreviation here>`
	
	Save any changes you make.

8. Edit the project settings:
    a. Open the `SAPAssetManager4.0Meter/SAM.mdkproject/MDKProject.json` file in a text editor. Edit app information such as:
        - Application name on the home screen
        - App version
        - Bundle ID to uniquely identify the application on the device
        - URL scheme for onboarding URLs
    b. Save any changes you make.
9. Set up the Mobile Development Kit Client SDK:
    a. Unzip the Mobile Development Kit SDK associated with the version you're building.
        1. Run the Mobile Development Kit dependencies installer and confirm that your system is ready.
        2. Unzip `MDKClient_SDK.zip` to the new `SAPAssetManager4.0Meter` folder.
    b. To install the necessary dependencies, open a Terminal prompt in the `SAPAssetManager4.0Meter/MDKClient_SDK` directory. Run either `./install.command` on a Mac or `./install.cmd` on a Windows PC. Note that you can build both iOS and an Android on a Mac. You can only build Android if you're using a Windows PC.

> **i Note**
>
> An internet connection is required. If you're connecting to the internet through a proxy, configure your settings before running the `./install.command` or `./install.cmd` command.

10. Create the SAP Asset Manager client:
    a. Open a Terminal prompt in the `SAPAssetManager4.0Meter/MDKClient_SDK` directory.
    b. Run the `create-client.command` command if you are on a Mac. Run the `create-client.cmd` command if you are on Windows.
    c. You can either specify command-line arguments to point to the `SAM.mdkproject` and the type of client (either device or simulator) you're building, or the script prompts you for the information.

> **Sample Code**
>
> on a Mac

```
$ ./create-client.command
? Enter the path of the .mdkproject directory. ../SAM.mdkproject
Using ../SAM.mdkproject
Using /Users/.../sdk for out directory
? Would you like to build for iOS or Android or All? ios
Building client for ios
? Would you like to build for device or simulator of iOS? device
Building client for device of iOS
Creating application SAPAssetManager4.0Meter
```

> **Sample Code**
>
> on a Windows PC

```
>create-client.cmd
? Enter the path of the .mdkproject directory. ..\SAM.mdkproject
Using ..\SAM.mdkproject
Using C:\...\mdk for out directory
Building client for Android
```

```
Creating application SAPAssetManager4.0Meter
```

d. Remove `demo.js` from the `MeterAssetManager/app` directory.

Results

After `create-client.command` for iOS or `create-client.cmd` for Windows finishes, you're ready to run the client either on the mobile device or on a simulator.

Next Steps

Open a Terminal prompt in the resulting client directory (ex: `SAPAssetManager4.0Meter/MDKClient_SDK/MeterAssetManager`). Run either the `tns run ios` or the `tns run android` command to start the application, based on your platform.

For iOS installations only, continue to the Allowing Custom URI Schemes [page 9] procedure.

3 SAP Asset Manager with Field Operations Worker

Field Operations Worker leverages the digital core with SAP S/4HANA for task driven activities and rounds.

Field Operations Worker supports workers who perform asset inspections and checks with focus on measurement points and on smaller services and repairs. Field Operations Worker adds the following functionality to the core SAP Asset Manager application:

- Display routes on a map: See a quick preview of your assigned routes from the Overview and Map pages
- View all route data: View information such as ID, description of the route, route priority, route due date, location, and number of stops on the route
- View all stop data: See stop location and technical objects at the stop
- Rapid field data capture: Allows you to take readings for all of the measuring points assigned to one technical object on a single screen

3.1 Building the Field Operations Worker Component Overview

Use the following information as a reference when building your application using the procedure Branding the Field Operations Worker Component for SAP Asset Manager [page 29].

Structure of .mdkproject

- **BrandedSettings.json:** Runtime configurations such as security settings, URLs for connecting to the SAP Cloud Platform mobile services, and more
- **MDKProject.json:** Build time configurations such as the application name, version, and bundle ID
- **App_Resources:** Any custom resources used by the application
- **demo:** To make an OData service available in demo mode, include the `.udb` and `.rq.udb` files for that service in this directory
- **extensions:** Include any extensions used by the application in this directory
- **metadata:** Built in metadata for the application

Configuring the MDKProject.json File

The `MDKProject.json` file contains settings that you can only configure before running the `create-client.command`:

- **AppName:** Determines the name of the application project and the app as it appears on an iOS device
- **AppVersion:** The client project application version
- **BaseProject:** The `metadata` subdirectory under the `.mdkproject` structure that contains the main application metadata. The main application metadata is the MDK application, which includes one or more component MDK applications. The component applications are only required if you are adding components to your base application, such as Meter Management or Field Operations Worker.
- **BundleID:** Uniquely identifies the resulting MDK client application on the device. Only one instance of a bundle ID can be installed on a device at a time. If you attempt to install a second application using the same bundle ID, it will overwrite the existing application.
- **Externals:** A list of NPM nodules that should not be included in the application bundle. Use this option for dependencies you expect to be in the environment when the application is built.
  Note that the modules `file-system` and `ui/dialogs` are automatically used as externals as they are already included in the client application.
- **URLScheme:** Allows you to specify a custom URL scheme that opens the client. If the URL includes connection settings such as URL parameters, these settings override the settings used by the client. Defaults to *mdkclient*.

Application Version and Notes on the Settings App

The Mobile Development Kit client tracks several versions, which you can view in the iOS *Settings* menu. These versions are identified as the *application* version, the *definitions* version, and the *frameworks* versions for the frameworks used in the client build.

When generating a client project, you can specify the application version. Specifying the application version allows you to version the client itself, which can be useful if you change extension controls or other branded settings. To specify the application version, specify the `AppVersion` property in the `MDKProject.json` file before running `create.client.command`.

To further customize the entry of your application in the iOS *Settings* menu, you can manually edit `<ProjectDirectory>/app/App_Resources/iOS/Settings.bundle/Root.plist` after the script has completed. You can add new entries, but do not remove existing entries or the application may not function correctly.

For more information, see Implementing an iOS Settings Bundle .

3.2 Branding the Field Operations Worker Component for SAP Asset Manager

Deploy the Field Operations Worker component from the out of the box configuration to set the cloud endpoint authentication URL and the OData service URL. You can also set other configuration values.

Prerequisites

- Verify that your system is set up to build the SAP Asset Manager application with the Field Operations Worker component by running the Mobile Development Kit Dependencies Installer tool. This tool detects all the components to install or update, allowing you to update or install them instantly.
  For more information and instructions on how to obtain the MDK Dependencies installer, see the Building Your MDK Client SDK procedure.
- Complete the Creating an Application in SAP Cloud Platform mobile services procedure.

Context

Use the following procedure to build and brand the SAP Asset Manager application with the Field Operations Worker component.

Procedure

1. Obtain the following SAP Asset Manager 1911 and Mobile Development Kit 4.0 installation files from the SAP Download Center:
   - Mobile Development Kit `MDK Plug-In SDK 4.0 folder`

     > **i Note**
     >
     > Select the following SAP Asset Manager and Mobile Development Kit versions based on your release:
     > - For SAP Asset Manager release 4.0.1, select Mobile Development Kit 3.1.3.
     > - For SAP Asset Manager release 4.0.2, select Mobile Development Kit 3.2.1.
     > - For SAP Asset Manager release 4.0.3, select Mobile Development Kit 3.2.4.
     >
     > The instructions in this procedure use the `MDK Plug-In SDK 4.0 folder` as an example, and may not be reflective of your installation.

   - Metadata 4.0 folder for SAP Asset Manager (`ASSET MANAGER METADATA 1911`)
   - Metadata 3.0 folder for Field Operations Worker (`FIELD OPS WORKER METADATA 4.0`)
   - Branding SDK for SAP Asset Manager, Meter Management, Field Operations Worker, Crew Management, Customer Service, and Asset Central (`ASSET MGR BRANDING SDK 1911`)

- Plug-ins: `MDK PLUG-IN SDK 4.0`

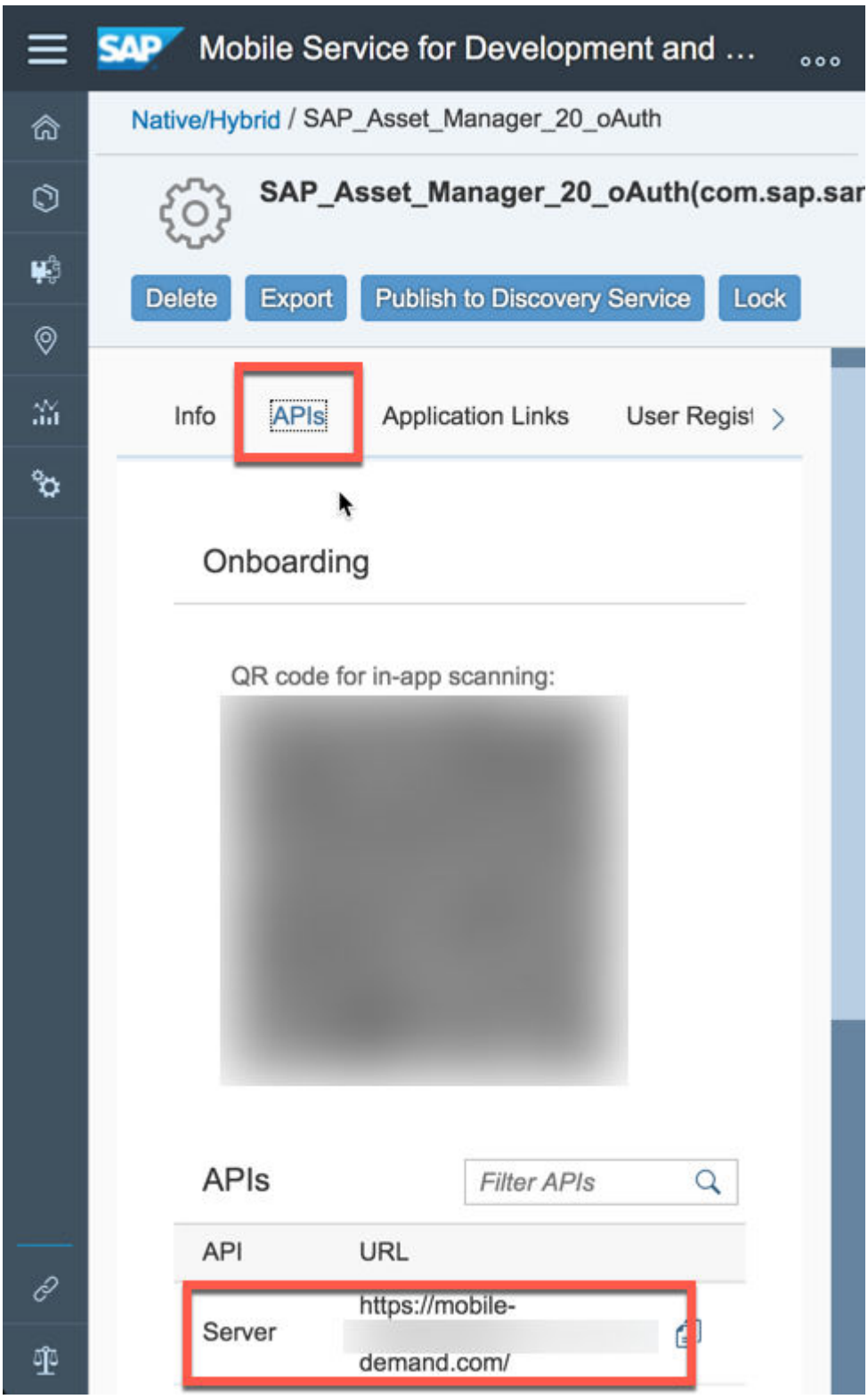
2. Create a folder to contain the installation files (`SAPAssetManager4.0FOW`).
3. Extract the SAP Asset Manager branding SDK:
   a. Unzip `ASSET MGR BRANDING SDK 1911`.
   b. Copy the `SAPAssetManagerFOW/SAM.mdkproject` folder to the `SAPAssetManager4.0FOW` folder.
   c. Set up the SAP Asset Manager Mobile Development Kit project folders:
      1. In the `SAPAssetManager4.0FOW/SAM.mdkproject/metadata` folder, create a folder named `fow.definitions`.
      2. In the `SAPAssetManager4.0FOW/SAM.mdkproject` folder, create a folder named `extensions`.
4. Extract the SAP Asset Manager metadata:
   a. Unzip the `ASSET MANAGER METADATA 1911` file.
   b. Copy all contents of the ZIP file to `SAPAssetManager4.0FOW/SAM.mdkproject/metadata/sam.definitions`.
   c. If present, delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0FOW/SAM.mdkproject/metadata/sam.definitions` folder.
5. Extract the Field Operations Worker SAP Asset Manager metadata:
   a. Unzip the `FIELD OPS WORKER METADATA 4.0` file.
   b. Copy all contents of the ZIP file to `SAPAssetManager4.0FOW/SAM.mdkproject/metadata/fow.definitions`.
   c. If present, delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0FOW/SAM.mdkproject/metadata/fow.definitions` folder.
6. Extract the MDK plug-in SDK in one of the following ways, based on whether you're building either an iOS or an Android client:

iOS Client	Android Client

1. **Unzip the iOS subfolder found in the `MDK PLUG-IN SDK 4.0` zip file.**
2. **Choose your architecture from one of the following folders:**
  ○ **Release-iphoneos**
  ○ **Release-iphonesimulator**
  ○ **Release-fat (contains both the iphoneos and the iphonesimulator architectures)**
3. **Copy the following folders from your selected architecture folder to the `SAPAssetManager4.0FOW/SAM.mdkproject/extensions` folder:**
  ○ `extension-Analytics`
  ○ `extension-BarcodeScanner`
  ○ `extension-FieldDataCapture`
  ○ `extension-MapFramework`
  ○ `extension-HierarchyFramework`
	1. Unzip the Android subfolder found in the `MDK PLUG-IN SDK 4.0` zip file.
	2. Copy the following folders from your `Universal` folder to the `SAPAssetManager4.0FOW/SAM.mdkproject/extensions` folder:
	  ○ `extension-Analytics`
	  ○ `extension-BarcodeScanner`
	  ○ `extension-FieldDataCapture`
	  ○ `extension-MapFramework`
	  ○ `extension-HierarchyFramework`

7. Configure the connection to SAP Cloud Platform mobile services:
    a. Retrieve the following information to establish a connection between the SAP Asset Manager application and the SAP Cloud Platform mobile services:
        - **AppId:** Note the ID under the Mobile Development Kit that you created in *Step 6* of the Creating an Application in SAP Cloud Platform mobile services procedure.
        - **ClientID:** Note the oAuth client ID that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

- **SapCloudPlatformEndpoint:** Find the Endpoint setting inside the application list of APIs on the *Mobile Services* under the *Server* API:



> **Note**
>
> By default, the *Server API* has a / at the end of the endpoint URL. Do not add this / into your connection settings.

- **AuthorizationEndpointURL:** Note the oAuth authorization endpoint URL that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **RedirectURL:** Note the callback URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **TokenURL:** Note the token URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

b. Choose your client configuration:

To preconfigure your client to connect to your mobile application, add the information retrieved in the above step to the *ConnectionSettings* block. When adding additional entries, include a comma after the existing *EnableOverrides* entry. An example is included in the table for reference when you're adding entries to your file.

If you want to use SAP Asset Manager against different back-end mobile applications (ex: DEV and QA), leave the *ConnectionSettings* as is, and build an onboarding URL for users using the values found in *Step 7a*. See the example in the table for further information on how to connect a client using either of the methods.

Preconfigured Client	Onboarding URL
You can generate a sample onboarding link to overwrite these values on a device with the following format in a URI:	You can generate an onboarding link to overwrite the values discussed in this substep on a device. Use the following format in a URI:
Sample Code … "DetailLabelViewText": "$(L,detail_label_view_text)", "ConnectionSettings": { "EnableOverrides": true, "AppId": "<Insert AppID value here>", "ClientId": "<Insert ClientID value here>", "SapCloudPlatformEndpoint": "https://<Insert cloud platform endpoint URL here>", "AuthorizationEndpointUrl": "https://<insert authorization endpoint URL here>", "RedirectUrl": "https://<insert redirect URL here>", "TokenUrl": "https://<insert token URL here>" },	**Sample Code** samclient://?AppId=<Insert AppID value here> &ClientId=<Insert ClientID value here> &SapCloudPlatformEndpoint=<Insert cloud platform endpoint URL here> &AuthorizationEndpointUrl=<insert authorization endpoint URL here> &RedirectUrl=<insert redirect URL here> &TokenUrl=<insert token URL here> &ServiceTimeZoneAbbreviation=<insert timezone abbreviation here>
Using the example as a guide, insert your own connection-specific values where they belong. **Save any changes you make.**	Save any changes you make.

8. Edit the project settings:
    a. Open the `SAPAssetManager4.0FOW/SAM.mdkproject/MDKProject.json` file in a text editor. Edit app information such as:
        - Application name on the home screen
        - App version
        - Bundle ID to uniquely identify the application on the device
        - URL scheme for onboarding URLs
    b. Save any changes you make.
9. Set up the Mobile Development Kit Client SDK:
    a. Unzip the Mobile Development Kit SDK associated with the version you're building.
        1. Run the Mobile Development Kit dependencies installer and confirm that your system is ready.
        2. Unzip `MDKClient_SDK.zip` to the new `SAPAssetManager4.0FOW` folder.
    b. To install the necessary dependencies, open a Terminal prompt in the `SAPAssetManager4.0Meter/MDKClient_SDK` directory. Run either `./install.command` on a Mac or `./install.cmd` on a Windows PC. Note that you can build both iOS and an Android on a Mac. You can only build Android if you're using a Windows PC.

> **i Note**
>
> An internet connection is required. If you're connecting to the internet through a proxy, configure your settings before running the `./install.command` or `./install.cmd` command.

10. Create the SAP Asset Manager client:
    a. Open a Terminal prompt in the `SAPAssetManager4.0FOW/MDKClient_SDK` directory.
    b. Run the `create-client.command` command if you are on a Mac. Run the `create-client.cmd` command if you are on Windows.
    c. You can either specify command-line arguments to point to the `SAM.mdkproject` and the type of client (either device or simulator) you are building, or the script prompts you for the information.

> **Sample Code**
>
> on a Mac

```
$ ./create-client.command
? Enter the path of the .mdkproject directory. ../SAM.mdkproject
Using ../SAM.mdkproject
Using /Users/.../sdk for out directory
? Would you like to build for iOS or Android or All? ios
Building client for ios
? Would you like to build for device or simulator of iOS? device
Building client for device of iOS
Creating application SAPAssetManager4.0FOW
```

> **Sample Code**
>
> on a Windows PC

```
>create-client.cmd
? Enter the path of the .mdkproject directory. ..\SAM.mdkproject
Using ..\SAM.mdkproject
Using C:\...\mdk for out directory
Building client for Android
```

```
Creating application SAPAssetManager4.0FOW
```

d. Remove `demo.js` from the `FOWAssetManager/app` directory.

Results

After `create-client.command` for iOS or `create-client.cmd` for Windows finishes, you're ready to run the client either on the mobile device or on a simulator.

Next Steps

Open a Terminal prompt in the resulting client directory (ex: `SAPAssetManager4.0FOW/MDKClient_SDK/MeterAssetManager`). Run either the `tns run ios` or the `tns run android` command to start the application, based on your platform.

For iOS installations only, continue to the Allowing Custom URI Schemes [page 9] procedure.

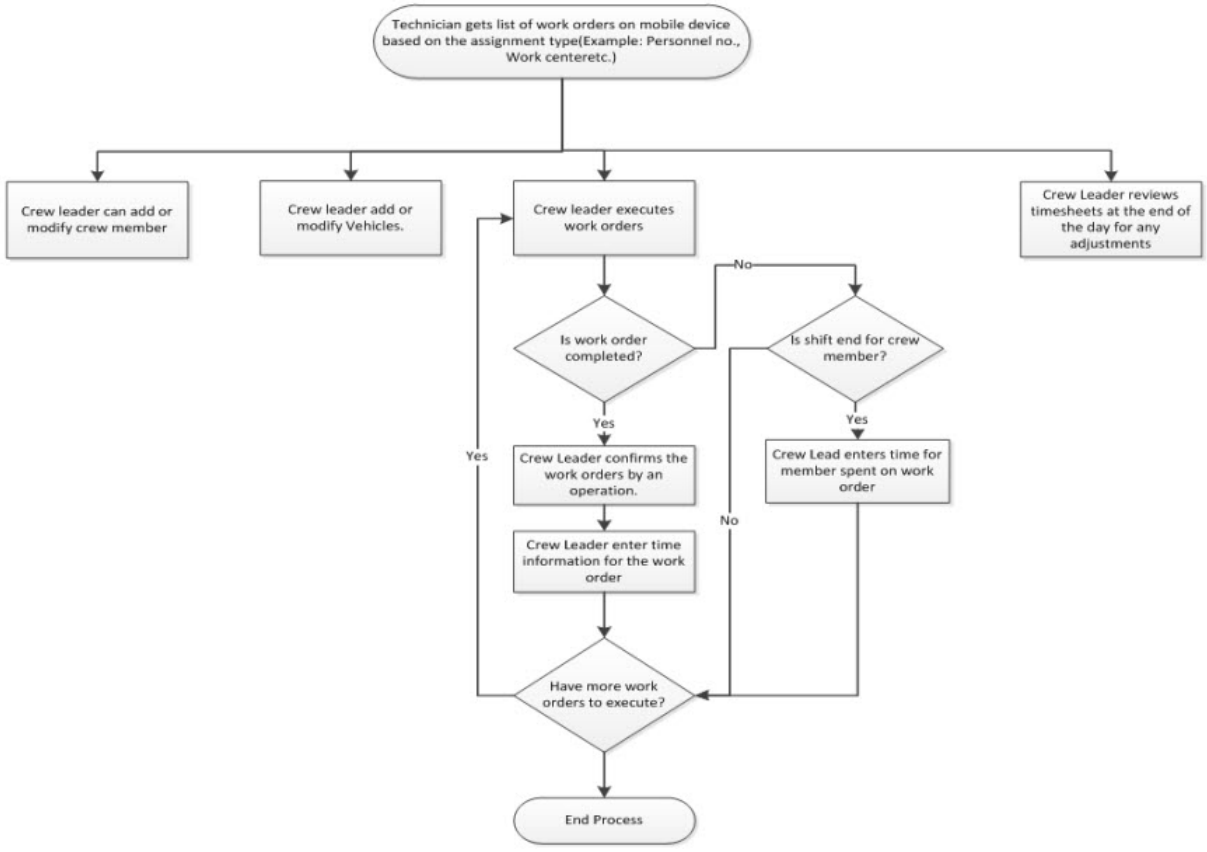
4 SAP Asset Manager with Crew Management

The Crew Management add-on component for SAP Asset Manager is an optional component that allows supervisors and team leaders to manage their crew with significantly lower cost and improved flexibility.

When the Crew Management add-on component is implemented, it adds the following functionality to the core SAP Asset Manager application:

- Add, remove, and select crew technicians
- Add, remove, and select vehicles
- Track vehicle usage through vehicle odometer readings
- Report and review time for crew

The following diagram is a sample of the workflow for Crew Management:



4.1 Building the Crew Management Component Overview

Use the following information as a reference when building your application using the procedure Branding the Crew Management Component for SAP Asset Manager [page 38].

Structure of .mdkproject

- **BrandedSettings.json:** Runtime configurations such as security settings, URLs for connecting to the SAP Cloud Platform mobile services, and more
- **MDKProject.json:** Build time configurations such as the application name, version, and bundle ID
- **App_Resources:** Any custom resources used by the application
- **demo:** To make an OData service available in demo mode, include the `.udb` and `.rq.udb` files for that service in this directory
- **extensions:** Include any extensions used by the application in this directory
- **metadata:** Built in metadata for the application

Configuring the MDKProject.json File

The `MDKProject.json` file contains settings that you can only configure before running the `create-client.command`:

- **AppName:** Determines the name of the application project and the app as it appears on an iOS device
- **AppVersion:** The client project application version
- **BaseProject:** The `metadata` subdirectory under the `.mdkproject` structure that contains the main application metadata. The main application metadata is the MDK application, which includes one or more component MDK applications. The component applications are only required if you are adding components to your base application, such as Meter Management or Field Operations Worker.
- **BundleID:** Uniquely identifies the resulting MDK client application on the device. Only one instance of a bundle ID can be installed on a device at a time. If you attempt to install a second application using the same bundle ID, it will overwrite the existing application.
- **Externals:** A list of NPM nodules that should not be included in the application bundle. Use this option for dependencies you expect to be in the environment when the application is built.
  Note that the modules `file-system` and `ui/dialogs` are automatically used as externals as they are already included in the client application.
- **URLScheme:** Allows you to specify a custom URL scheme that opens the client. If the URL includes connection settings such as URL parameters, these settings override the settings used by the client. Defaults to *mdkclient*.

Application Version and Notes on the Settings App

The Mobile Development Kit client tracks several versions, which you can view in the iOS *Settings* menu. These versions are identified as the *application* version, the *definitions* version, and the *frameworks* versions for the frameworks used in the client build.

When generating a client project, you can specify the application version. Specifying the application version allows you to version the client itself, which can be useful if you change extension controls or other branded settings. To specify the application version, specify the `AppVersion` property in the `MDKProject.json` file before running `create.client.command`.

To further customize the entry of your application in the iOS *Settings* menu, you can manually edit `<ProjectDirectory>/app/App_Resources/iOS/Settings.bundle/Root.plist` after the script has completed. You can add new entries, but do not remove existing entries or the application may not function correctly.

For more information, see Implementing an iOS Settings Bundle .

4.2 Branding the Crew Management Component for SAP Asset Manager

Deploy the Crew Management component from the out of the box configuration to set the cloud endpoint authentication URL and the OData service URL. You can also set other configuration values.

Prerequisites

- Verify that your system is set up to build the SAP Asset Manager application with the Crew Management component by running the Mobile Development Kit Dependencies Installer tool. This tool detects all the components to install or update, allowing you to update or install them instantly.
  For more information and instructions on how to obtain the MDK Dependencies installer, see the Building Your MDK Client SDK procedure.
- Complete the Creating an Application in SAP Cloud Platform mobile services procedure.

Context

Use the following procedure to build and brand the SAP Asset Manager application with the Crew Management component.

Procedure

1. Obtain the following SAP Asset Manager 1911 and Mobile Development Kit 4.0 installation files from the SAP Download Center:
   - Mobile Development Kit `MDK Plug-In SDK 4.0 folder`

> **i Note**
>
> Select the following SAP Asset Manager and Mobile Development Kit versions based on your release:
> - For SAP Asset Manager release 4.0.1, select Mobile Development Kit 3.1.3.
> - For SAP Asset Manager release 4.0.2, select Mobile Development Kit 3.2.1.
> - For SAP Asset Manager release 4.0.3, select Mobile Development Kit 3.2.4.
>
> The instructions in this procedure use the `MDK Plug-In SDK 4.0 folder` as an example, and may not be reflective of your installation.

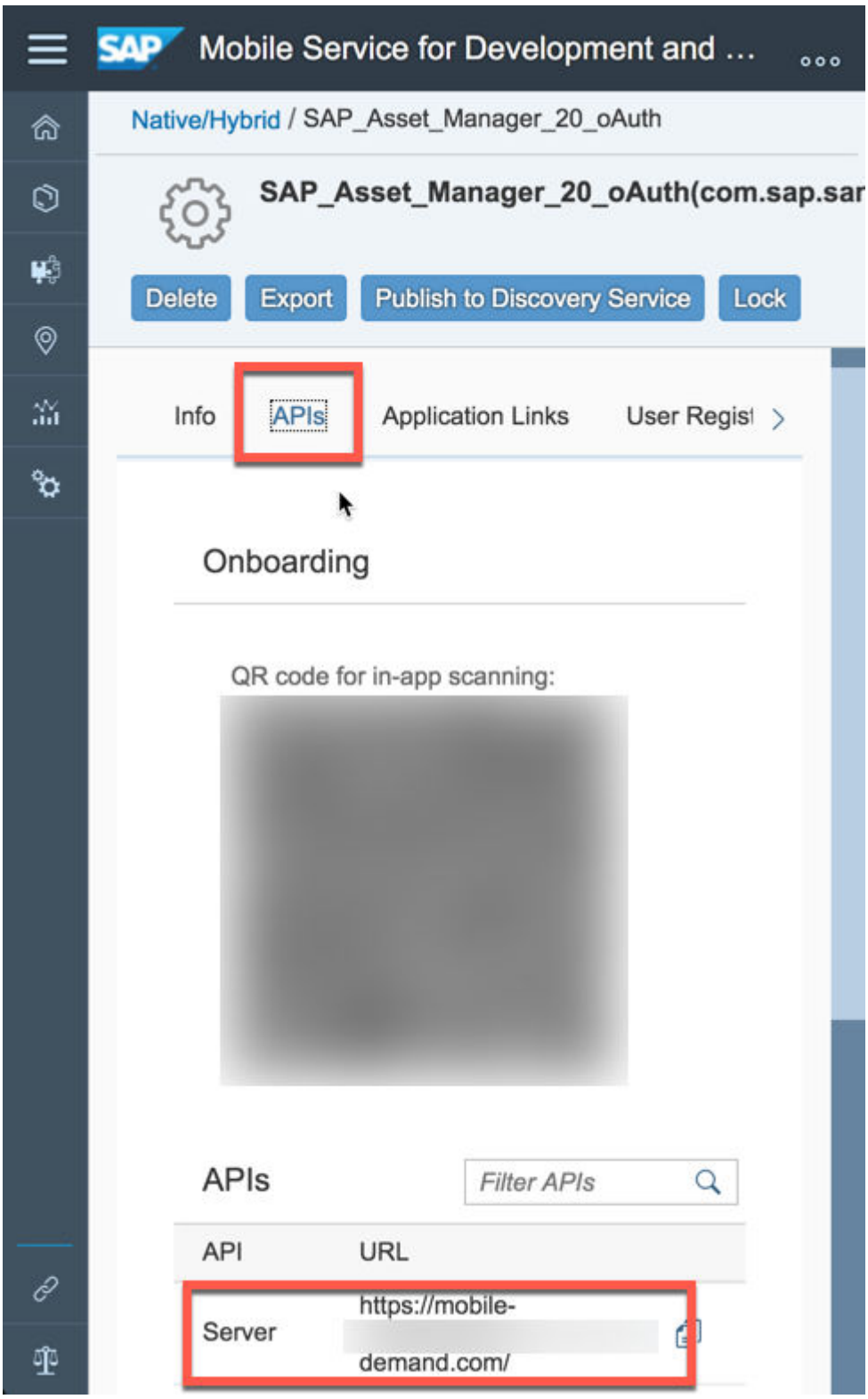
   - Metadata 4.0 folder for (`ASSET MANAGER METADATA 1911`)
   - Metadata 3.0 folder for Crew Management (`ASSET MGR CREW METADATA 4.0`)
   - Branding SDK for SAP Asset Manager, Meter Management, Field Operations Worker, Crew Management, Customer Service, and Asset Central (`ASSET MGR BRANDING SDK 1911`)
   - Plug-ins: `MDK PLUG-IN SDK 4.0`
2. Create a folder to contain the installation files (`SAPAssetManager4.0Crew`).
3. Extract the SAP Asset Manager branding SDK:
   a. Unzip `ASSET MGR BRANDING SDK 1911`.
   b. Copy the `SAPAssetManagerCrew/SAM.mdkproject` folder to the `SAPAssetManager4.0Crew` folder.
   c. Set up the SAP Asset Manager Mobile Development Kit project folders:
      1. In the `SAPAssetManager4.0Crew/SAM.mdkproject/metadata` folder, create a folder named `crew.definitions`.
      2. In the `SAPAssetManager4.0Crew/SAM.mdkproject` folder, create a folder named `extensions`.
4. Extract the SAP Asset Manager metadata:
   a. Unzip the `ASSET MANAGER METADATA 1911` file.
   b. Copy all contents of the ZIP file to folder, create a folder named `SAPAssetManager4.0Crew/SAM.mdkproject/metadata/sam.definitions`.
   c. If present, delete the `SIGNATURE.SMF` folder, create a file from the `SAPAssetManager4.0Crew/SAM.mdkproject/metadata/sam.definitions` folder.
5. Extract the Crew Management SAP Asset Manager metadata:
   a. Unzip the `ASSET MGR CREW METADATA 4.0` file.
   b. Copy all contents of the ZIP file to `SAPAssetManager4.0Crew/SAM.mdkproject/metadata/crew.definitions`.
   c. If present, delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0Crew/SAM.mdkproject/metadata/crew.definitions` folder.
6. Extract the MDK plug-in SDK in one of the following ways, based on whether you're building either an iOS or an Android client:

iOS Client	Android Client
1. **Unzip the iOS subfolder found in the** `MDK PLUG-IN SDK 4.0` **zip file.**	1. Unzip the Android subfolder found in the `MDK PLUG-IN SDK 4.0` zip file.

iOS Client	Android Client
2. **Choose your architecture from one of the following folders:** ○ **Release-iphoneos** ○ **Release-iphonesimulator** ○ **Release-fat (contains both the iphoneos and the iphonesimulator architectures)** 3. **Copy the following folders from your selected architecture folder to the `SAPAssetManager4.0Crew/SAM.mdkproject/extensions` folder:** ○ `extension-Analytics` ○ `extension-BarcodeScanner` ○ `extension-FieldDataCapture` ○ `extension-MapFramework` ○ `extension-HierarchyFramework`	2. Copy the following folders from your `Universal` folder to the `SAPAssetManager4.0Crew/SAM.mdkproject/extensions` folder: ○ `extension-Analytics` ○ `extension-BarcodeScanner` ○ `extension-FieldDataCapture` ○ `extension-MapFramework` ○ `extension-HierarchyFramework`

7. Configure the connection to SAP Cloud Platform mobile services:
   a. Retrieve the following information to establish a connection between the SAP Asset Manager application and the SAP Cloud Platform mobile services:
      - **AppId:** Note the ID under the Mobile Development Kit that you created in *Step 6* of the Creating an Application in SAP Cloud Platform mobile services procedure.
      - **ClientID:** Note the oAuth client ID that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

- **SapCloudPlatformEndpoint:** Find the Endpoint setting inside the application list of APIs on the *Mobile Services* under the *Server* API:



> **Note**
>
> By default, the *Server API* has a / at the end of the endpoint URL. Do not add this / into your connection settings.

- **AuthorizationEndpointURL:** Note the oAuth authorization endpoint URL that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **RedirectURL:** Note the callback URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **TokenURL:** in a Note the token URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

b. Choose your client configuration:

To preconfigure your client to connect to your mobile application, add the information retrieved in the above step to the *ConnectionSettings* block. When adding additional entries, include a comma after the existing *EnableOverrides* entry. An example is included in the table for reference when you're adding entries to your file.

If you want to use SAP Asset Manager against different back-end mobile applications (ex: DEV and QA), leave the *ConnectionSettings* as is, and build an onboarding URL for users using the values found in *Step 7a*. See the example in the table for further information on how to connect a client using either of the methods.

Preconfigured Client	Onboarding URL
You can generate a sample onboarding link to overwrite these values on a device with the following format in a URI:	You can generate an onboarding link to overwrite the values discussed in this substep on a device. Use the following format in a URI:
Sample Code … "DetailLabelViewText": "$(L,detail_label_view_text)", "ConnectionSettings": { "EnableOverrides": true, "AppId": "<Insert AppID value here>", "ClientId": "<Insert ClientID value here>", "SapCloudPlatformEndpoint": "https://<Insert cloud platform endpoint URL here>", "AuthorizationEndpointUrl": "https://<insert authorization endpoint URL here>", "RedirectUrl": "https://<insert redirect URL here>", "TokenUrl": "https://<insert token URL here>" },	**Sample Code** samclient://?AppId=<Insert AppID value here> &ClientId=<Insert ClientID value here> &SapCloudPlatformEndpoint=<Insert cloud platform endpoint URL here> &AuthorizationEndpointUrl=<insert authorization endpoint URL here> &RedirectUrl=<insert redirect URL here> &TokenUrl=<insert token URL here> &ServiceTimeZoneAbbreviation=<insert timezone abbreviation here>
Using the example as a guide, insert your own connection-specific values where they belong. **Save any changes you make.**	Save any changes you make.

8. Edit the project settings:
    a. Open the `SAPAssetManager4.0Crew/SAM.mdkproject/MDKProject.json` file in a text editor. Edit app information such as:
        - Application name on the home screen
        - App version
        - Bundle ID to uniquely identify the application on the device
        - URL scheme for onboarding URLs
    b. Save any changes you make.
9. Set up the Mobile Development Kit Client SDK:
    a. Unzip the Mobile Development Kit SDK associated with the version you're building.
        1. Run the Mobile Development Kit dependencies installer and confirm that your system is ready.
        2. Unzip `MDKClient_SDK.zip` to the new `SAPAssetManager4.0Crew` folder.
    b. To install the necessary dependencies, open a Terminal prompt in the `SAPAssetManager4.0Meter/MDKClient_SDK` directory. Run either `./install.command` on a Mac or `./install.cmd` on a Windows PC. Note that you can build both iOS and an Android on a Mac. You can only build Android if you're using a Windows PC.

> **i Note**
>
> An internet connection is required. If you're connecting to the internet through a proxy, configure your settings before running the `./install.command` or `./install.cmd` command.

10. Create the SAP Asset Manager client:
    a. Open a Terminal prompt in the `SAPAssetManager3.0Crew/MDKClient_SDK` directory.
    b. Run the `create-client.command` command if you are on a Mac. Run the `create-client.cmd` command if you are on Windows.
    c. You can either specify command-line arguments to point to the `SAM.mdkproject` and the type of client (either device or simulator) you are building, or the script prompts you for the information.

> **Sample Code**
>
> on a Mac

```
$ ./create-client.command
? Enter the path of the .mdkproject directory. ../SAM.mdkproject
Using ../SAM.mdkproject
Using /Users/.../sdk for out directory
? Would you like to build for iOS or Android or All? ios
Building client for ios
? Would you like to build for device or simulator of iOS? device
Building client for device of iOS
Creating application SAPAssetManager3.0Crew
```

> **Sample Code**
>
> on a Windows PC

```
>create-client.cmd
? Enter the path of the .mdkproject directory. ..\SAM.mdkproject
Using ..\SAM.mdkproject
Using C:\...\mdk for out directory
Building client for Android
```

```
Creating application SAPAssetManager3.0Crew
```

d. Remove `demo.js` from the `CrewAssetManager/app` directory.

Results

After `create-client.command` for iOS or `create-client.cmd` for Windows finishes, you're ready to run the client either on the mobile device or on a simulator.

Next Steps

Open a Terminal prompt in the resulting client directory (ex: `SAPAssetManager4.0Crew/MDKClient_SDK/MeterAssetManager`). Run either the `tns run ios` or the `tns run android` command to start the application, based on your platform.

For iOS installations only, continue to the Allowing Custom URI Schemes [page 9] procedure.

5 SAP Asset Manager with Customer Service

The Customer Service is used at sites where work on the application involves maintenance or services performed under contract for customers or other third parties.

Customer Service adds the following functionality to the core SAP Asset Manager application:

- Details of service engagements of the technician with the customer
- Details of business customer information associated with the customer
- Details of contract and warranty information for the customer
- Online mapping functionality for both customer addresses and partner address

5.1 Building the Customer Service Component Overview

Use the following information as a reference when building your application using the procedure Branding the Customer Service Component for SAP Asset Manager [page 46].

Structure of .mdkproject

- **BrandedSettings.json:** Runtime configurations such as security settings, URLs for connecting to the SAP Cloud Platform mobile services, and more
- **MDKProject.json:** Build time configurations such as the application name, version, and bundle ID
- **App_Resources:** Any custom resources used by the application
- **demo:** To make an OData service available in demo mode, include the `.udb` and `.rq.udb` files for that service in this directory
- **extensions:** Include any extensions used by the application in this directory
- **metadata:** Built in metadata for the application

Configuring the MDKProject.json File

The `MDKProject.json` file contains settings that you can only configure before running the `create-client.command`:

- **AppName:** Determines the name of the application project and the app as it appears on an iOS device
- **AppVersion:** The client project application version
- **BaseProject:** The `metadata` subdirectory under the `.mdkproject` structure that contains the main application metadata. The main application metadata is the MDK application, which includes one or more

component MDK applications. The component applications are only required if you are adding components to your base application, such as Meter Management or Field Operations Worker.

- **BundleID:** Uniquely identifies the resulting MDK client application on the device. Only one instance of a bundle ID can be installed on a device at a time. If you attempt to install a second application using the same bundle ID, it will overwrite the existing application.
- **Externals:** A list of NPM nodules that should not be included in the application bundle. Use this option for dependencies you expect to be in the environment when the application is built.
  Note that the modules `file-system` and `ui/dialogs` are automatically used as externals as they are already included in the client application.
- **URLScheme:** Allows you to specify a custom URL scheme that opens the client. If the URL includes connection settings such as URL parameters, these settings override the settings used by the client. Defaults to *mdkclient*.

Application Version and Notes on the Settings App

The Mobile Development Kit client tracks several versions, which you can view in the iOS *Settings* menu. These versions are identified as the *application* version, the *definitions* version, and the *frameworks* versions for the frameworks used in the client build.

When generating a client project, you can specify the application version. Specifying the application version allows you to version the client itself, which can be useful if you change extension controls or other branded settings. To specify the application version, specify the `AppVersion` property in the `MDKProject.json` file before running `create.client.command`.

To further customize the entry of your application in the iOS *Settings* menu, you can manually edit `<ProjectDirectory>/app/App_Resources/iOS/Settings.bundle/Root.plist` after the script has completed. You can add new entries, but do not remove existing entries or the application may not function correctly.

For more information, see Implementing an iOS Settings Bundle .

5.2 Branding the Customer Service Component for SAP Asset Manager

Deploy the Customer Service component from the out of the box configuration to set the cloud endpoint authentication URL and the OData service URL. You can also set other configuration values.

Prerequisites

- Verify that your system is set up to build the SAP Asset Manager application with the Customer Service component by running the Mobile Development Kit Dependencies Installer tool. This tool detects all the components to install or update, allowing you to update or install them instantly.
  For more information and instructions on how to obtain the MDK Dependencies installer, see the Building Your MDK Client SDK procedure.
- Complete the Creating an Application in SAP Cloud Platform mobile services procedure.

Context

Use the following procedure to build and brand the SAP Asset Manager application with the Customer Service component.

Procedure

1. Obtain the following SAP Asset Manager 1911 and Mobile Development Kit 4.0 installation files from the SAP Download Center:
   - Mobile Development Kit `MDK Plug-In SDK 4.0 folder`

     > **i Note**
     >
     > Select the following SAP Asset Manager and Mobile Development Kit versions based on your release:
     > - For SAP Asset Manager release 4.0.1, select Mobile Development Kit 3.1.3.
     > - For SAP Asset Manager release 4.0.2, select Mobile Development Kit 3.2.1.
     > - For SAP Asset Manager release 4.0.3, select Mobile Development Kit 3.2.4.
     >
     > The instructions in this procedure use the `MDK Plug-In SDK 4.0 folder` as an example, and may not be reflective of your installation.

   - Metadata 4.0 folder for SAP Asset Manager (`ASSET MANAGER METADATA 1911`)
   - Metadata 2.0 folder for Customer Service (`ASSET MGR CUSTOMERSVC META 3.0`)
   - Branding SDK for SAP Asset Manager, Meter Management, Field Operations Worker, Crew Management, Customer Service, and Asset Central (`ASSET MGR BRANDING SDK 1911`)
   - Plug-ins: `MDK PLUG-IN SDK 4.0`
2. Create a folder to contain the installation files (`SAPAssetManager4.0Customer`).
3. Extract the SAP Asset Manager branding SDK:
   a. Unzip `ASSET MGR BRANDING SDK 1911`.
   b. Copy the `SAPAssetManagerCustomer/SAM.mdkproject` folder to the `SAPAssetManager4.0Customer` folder.
   c. Set up the SAP Asset Manager Mobile Development Kit project folders:
      1. In the `SAPAssetManager4.0Customer/SAM.mdkproject/metadata` folder, create a folder named `customer.definitions`.
      2. In the `SAPAssetManager4.0Customer/SAM.mdkproject` folder, create a folder named `extensions`.
4. Extract the SAP Asset Manager metadata:
   a. Unzip the `ASSET MANAGER METADATA 1911` file.
   b. Copy all contents of the ZIP file to `SAPAssetManager4.0Customer/SAM.mdkproject/metadata/sam.definitions`.
   c. If present, delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0Customer/SAM.mdkproject/metadata/sam.definitions` folder.
5. Extract the Customer Service SAP Asset Manager metadata:

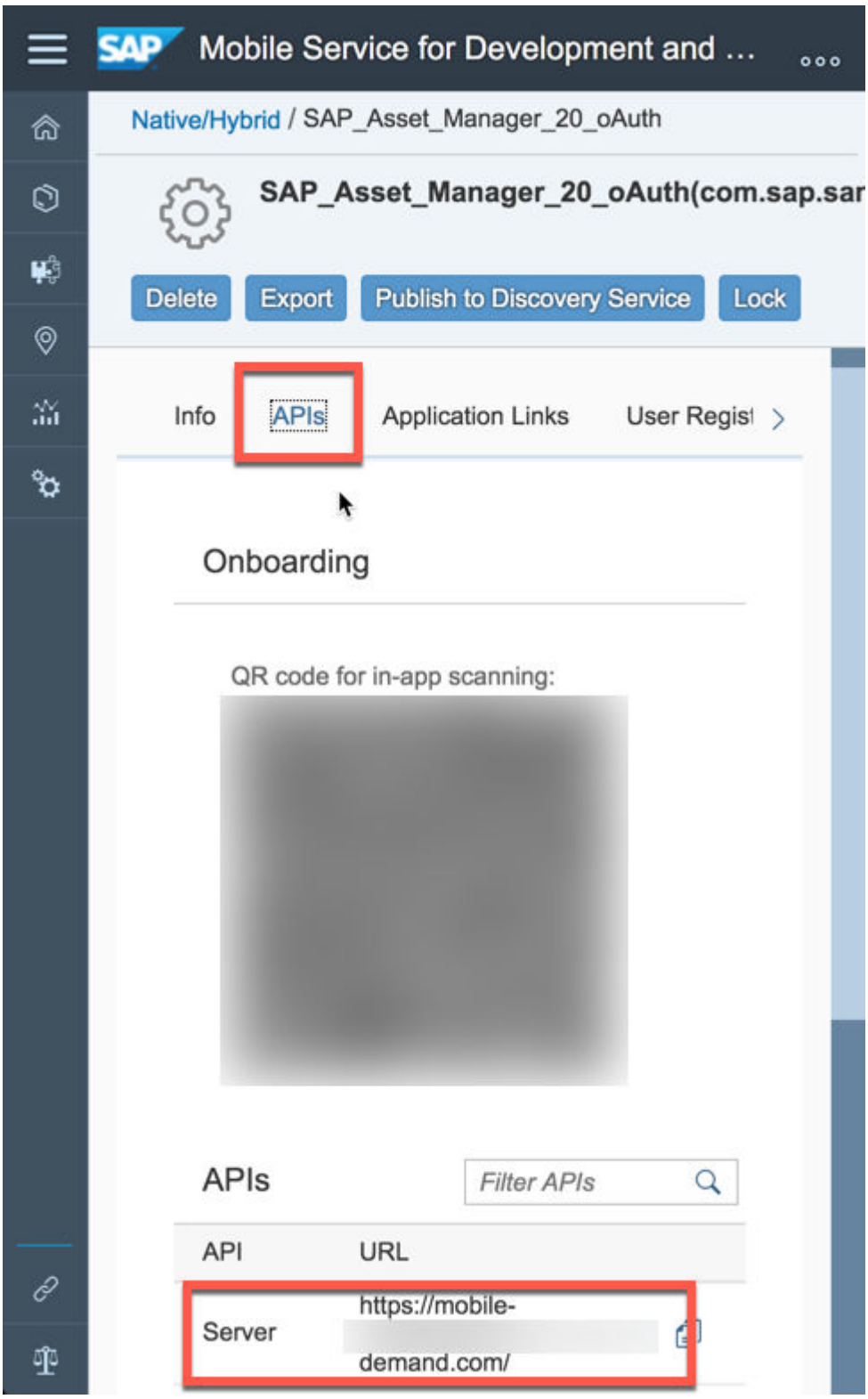
a. Unzip the `ASSET MGR CUSTOMERSVC META 3.0` file.
b. Copy all contents of the ZIP file to `SAPAssetManager4.0Customer/SAM.mdkproject/metadata/customer.definitions`.
c. If present, delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0Customer/SAM.mdkproject/metadata/customer.definitions` folder.

6. Extract the MDK plug-in SDK in one of the following ways, based on whether you're building either an iOS or an Android client:

iOS Client	Android Client
1. **Unzip the iOS subfolder found in the `MDK PLUG-IN SDK 4.0` zip file.** 2. **Choose your architecture from one of the following folders:** ○ **Release-iphoneos** ○ **Release-iphonesimulator** ○ **Release-fat (contains both the iphoneos and the iphonesimulator architectures)** 3. **Copy the following folders from your selected architecture folder to the `SAPAssetManager4.0Customer/SAM.mdkproject/extensions` folder:** ○ `extension-Analytics` ○ `extension-BarcodeScanner` ○ `extension-FieldDataCapture` ○ `extension-MapFramework` ○ `extension-HierarchyFramework`	1. Unzip the Android subfolder found in the `MDK PLUG-IN SDK 4.0` zip file. 2. Copy the following folders from your `Universal` folder to the `SAPAssetManager4.0Customer/SAM.mdkproject/extensions` folder: ○ `extension-Analytics` ○ `extension-BarcodeScanner` ○ `extension-FieldDataCapture` ○ `extension-MapFramework` ○ `extension-HierarchyFramework`

7. Configure the connection to SAP Cloud Platform mobile services:
   a. Retrieve the following information to establish a connection between the SAP Asset Manager application and the SAP Cloud Platform mobile services:
      ○ **AppId:** Note the ID under the Mobile Development Kit that you created in *Step 6* of the Creating an Application in SAP Cloud Platform mobile services procedure.
      ○ **ClientID:** Note the oAuth client ID that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

- **SapCloudPlatformEndpoint:** Find the Endpoint setting inside the application list of APIs on the *Mobile Services* under the *Server* API:



> **Note**
>
> By default, the *Server API* has a / at the end of the endpoint URL. Do not add this / into your connection settings.

- **AuthorizationEndpointURL:** Note the oAuth authorization endpoint URL that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **RedirectURL:** Note the callback URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **TokenURL:** Note the token URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

b. Choose your client configuration:

To preconfigure your client to connect to your mobile application, add the information retrieved in the above step to the *ConnectionSettings* block. When adding additional entries, include a comma after the existing *EnableOverrides* entry. An example is included in the table for reference when you're adding entries to your file.

If you want to use SAP Asset Manager against different back-end mobile applications (ex: DEV and QA), leave the *ConnectionSettings* as is, and build an onboarding URL for users using the values found in *Step 7a*. See the example in the table for further information on how to connect a client using either of the methods.

Preconfigured Client	Onboarding URL
You can generate a sample onboarding link to overwrite these values on a device with the following format in a URI:	You can generate an onboarding link to overwrite the values discussed in this substep on a device. Use the following format in a URI:
Sample Code … "DetailLabelViewText": "$(L,detail_label_view_text)", "ConnectionSettings": { "EnableOverrides": true, "AppId": "<Insert AppID value here>", "ClientId": "<Insert ClientID value here>", "SapCloudPlatformEndpoint": "https://<Insert cloud platform endpoint URL here>", "AuthorizationEndpointUrl": "https://<insert authorization endpoint URL here>", "RedirectUrl": "https://<insert redirect URL here>", "TokenUrl": "https://<insert token URL here>" },	**Sample Code** samclient://?AppId=<Insert AppID value here> &ClientId=<Insert ClientID value here> &SapCloudPlatformEndpoint=<Insert cloud platform endpoint URL here> &AuthorizationEndpointUrl=<insert authorization endpoint URL here> &RedirectUrl=<insert redirect URL here> &TokenUrl=<insert token URL here> &ServiceTimeZoneAbbreviation=<insert timezone abbreviation here>
Using the example as a guide, insert your own connection-specific values where they belong. **Save any changes you make.**	Save any changes you make.

8. Edit the project settings:
    a. Open the `SAPAssetManager4.0Customer/SAM.mdkproject/MDKProject.json` file in a text editor. Edit app information such as:
        - Application name on the home screen
        - App version
        - Bundle ID to uniquely identify the application on the device
        - URL scheme for onboarding URLs
    b. Save any changes you make.
9. Set up the Mobile Development Kit Client SDK:
    a. Unzip the Mobile Development Kit SDK associated with the version you're building.
        1. Run the Mobile Development Kit dependencies installer and confirm that your system is ready.
        2. Unzip `MDKClient_SDK.zip` to the new `SAPAssetManager4.0Customer` folder.
    b. To install the necessary dependencies, open a Terminal prompt in the `SAPAssetManager4.0Meter/MDKClient_SDK` directory. Run either `./install.command` on a Mac or `./install.cmd` on a Windows PC. Note that you can build both iOS and an Android on a Mac. You can only build Android if you're using a Windows PC.

> **i Note**
>
> An internet connection is required. If you're connecting to the internet through a proxy, configure your settings before running the `./install.command` or `./install.cmd` command.

10. Create the SAP Asset Manager client:
    a. Open a Terminal prompt in the `SAPAssetManager4.0Customer/MDKClient_SDK` directory.
    b. Run the `create-client.command` command if you are on a Mac. Run the `create-client.cmd` command if you are on Windows.
    c. You can either specify command-line arguments to point to the `SAM.mdkproject` and the type of client (either device or simulator) you are building, or the script prompts you for the information.

> **Sample Code**
>
> on a Mac

```
$ ./create-client.command
? Enter the path of the .mdkproject directory. ../SAM.mdkproject
Using ../SAM.mdkproject
Using /Users/.../sdk for out directory
? Would you like to build for iOS or Android or All? ios
Building client for ios
? Would you like to build for device or simulator of iOS? device
Building client for device of iOS
Creating application SAPAssetManager4.0Customer
```

> **Sample Code**
>
> on a Windows PC

```
>create-client.cmd
? Enter the path of the .mdkproject directory. ..\SAM.mdkproject
Using ..\SAM.mdkproject
Using C:\...\mdk for out directory
Building client for Android
```

```
Creating application SAPAssetManager4.0Customer
```

d. Remove `demo.js` from the `CustomerAssetManager/app` directory.

Results

After `create-client.command` for iOS or `create-client.cmd` for Windows finishes, you're ready to run the client either on the mobile device or on a simulator.

Next Steps

Open a Terminal prompt in the resulting client directory (ex: `SAPAssetManager4.0Customer/MDKClient_SDK/MeterAssetManager`). Run either the `tns run ios` or the `tns run android` command to start the application, based on your platform.

For iOS installations only, continue to the Allowing Custom URI Schemes [page 9] procedure.

6 SAP Asset Manager with Asset Central

Asset Central links production systems and assets with manufacturing and maintenance business processes to reduce operational and maintenance costs and increase asset uptime.

Asset Central communicates with the Asset Intelligence Network back end. The Asset Intelligence Network is an SAP S/4HANA system.

The Asset Intelligence Network collects and tracks equipment information in a central repository. It facilitates collaborative asset management and lets you take full advantage of the Internet of Things (IoT). Organizations manage thousands of assets to keep their plants operational, which include machinery equipment. Timely maintenance is required to ensure that the equipment is working at an optimum level. Maintenance can be difficult due to heterogenous systems, missing information, or other challenges. The Asset Intelligence Network can help to solve these issues.

The SAP Asset Manager application uses PdMS indicators. PdMS, or Predictive Maintenance and Service equipment indicators, allow you to easily identify the health status of your equipment. PdMS is part of the Asset Central component, which relays information to and from the Asset Intelligence Network back end.

6.1 Building the Asset Central Component Overview

Use the following information as a reference when building your application using the procedure Branding the Asset Central Component for SAP Asset Manager [page 55].

Structure of .mdkproject

- **BrandedSettings.json:** Runtime configurations such as security settings, URLs for connecting to the SAP Cloud Platform mobile services, and more
- **MDKProject.json:** Build time configurations such as the application name, version, and bundle ID
- **App_Resources:** Any custom resources used by the application
- **demo:** To make an OData service available in demo mode, include the `.udb` and `.rq.udb` files for that service in this directory
- **extensions:** Include any extensions used by the application in this directory
- **metadata:** Built in metadata for the application

Configuring the MDKProject.json File

The `MDKProject.json` file contains settings that you can only configure before running the `create-client.command`:

- **AppName:** Determines the name of the application project and the app as it appears on an iOS device
- **AppVersion:** The client project application version
- **BaseProject:** The `metadata` subdirectory under the `.mdkproject` structure that contains the main application metadata. The main application metadata is the MDK application, which includes one or more component MDK applications. The component applications are only required if you are adding components to your base application, such as Meter Management or Field Operations Worker.
- **BundleID:** Uniquely identifies the resulting MDK client application on the device. Only one instance of a bundle ID can be installed on a device at a time. If you attempt to install a second application using the same bundle ID, it will overwrite the existing application.
- **Externals:** A list of NPM nodules that should not be included in the application bundle. Use this option for dependencies you expect to be in the environment when the application is built.
  Note that the modules `file-system` and `ui/dialogs` are automatically used as externals as they are already included in the client application.
- **URLScheme:** Allows you to specify a custom URL scheme that opens the client. If the URL includes connection settings such as URL parameters, these settings override the settings used by the client. Defaults to *mdkclient*.

Application Version and Notes on the Settings App

The Mobile Development Kit client tracks several versions, which you can view in the iOS *Settings* menu. These versions are identified as the *application* version, the *definitions* version, and the *frameworks* versions for the frameworks used in the client build.

When generating a client project, you can specify the application version. Specifying the application version allows you to version the client itself, which can be useful if you change extension controls or other branded settings. To specify the application version, specify the `AppVersion` property in the `MDKProject.json` file before running `create.client.command`.

To further customize the entry of your application in the iOS *Settings* menu, you can manually edit `<ProjectDirectory>/app/App_Resources/iOS/Settings.bundle/Root.plist` after the script has completed. You can add new entries, but do not remove existing entries or the application may not function correctly.

For more information, see Implementing an iOS Settings Bundle .

6.2 Branding the Asset Central Component for SAP Asset Manager

Deploy the Asset Central component from the out of the box configuration to set the cloud endpoint authentication URL and the OData service URL. You can also set other configuration values.

Prerequisites

- Verify that your system is set up to build the SAP Asset Manager application with the Asset Central component by running the Mobile Development Kit Dependencies Installer tool. This tool detects all the components to install or update, allowing you to update or install them instantly.
  For more information and instructions on how to obtain the MDK Dependencies installer, see the Building Your MDK Client SDK procedure.
- Complete the Creating an Application in SAP Cloud Platform mobile services procedure.

Context

Use the following procedure to build and brand the SAP Asset Manager application with the Asset Central component.

Procedure

1. Obtain the following SAP Asset Manager 1911 and Mobile Development Kit 4.0 installation files from the SAP Download Center:
   - Mobile Development Kit `MDK Plug-In SDK 4.0 folder`

     > **i Note**
     >
     > Select the following SAP Asset Manager and Mobile Development Kit versions based on your release:
     > - For SAP Asset Manager release 4.0.1, select Mobile Development Kit 3.1.3.
     > - For SAP Asset Manager release 4.0.2, select Mobile Development Kit 3.2.1.
     > - For SAP Asset Manager release 4.0.3, select Mobile Development Kit 3.2.4.
     >
     > The instructions in this procedure use the `MDK Plug-In SDK 4.0 folder` as an example, and may not be reflective of your installation.

   - Metadata 4.0 folder for SAP Asset Manager (`ASSET MANAGER METADATA 1911`)
   - Metadata 2.0 folder for Asset Central (`ASSETMGR ASSETCENTRAL META 2.0`)
   - Branding SDK for SAP Asset Manager, Meter Management, Field Operations Worker, Crew Management, Customer Service, and Asset Central (`ASSET MGR BRANDING SDK 1911`)

- Plug-ins: `MDK PLUG-IN SDK 4.0`

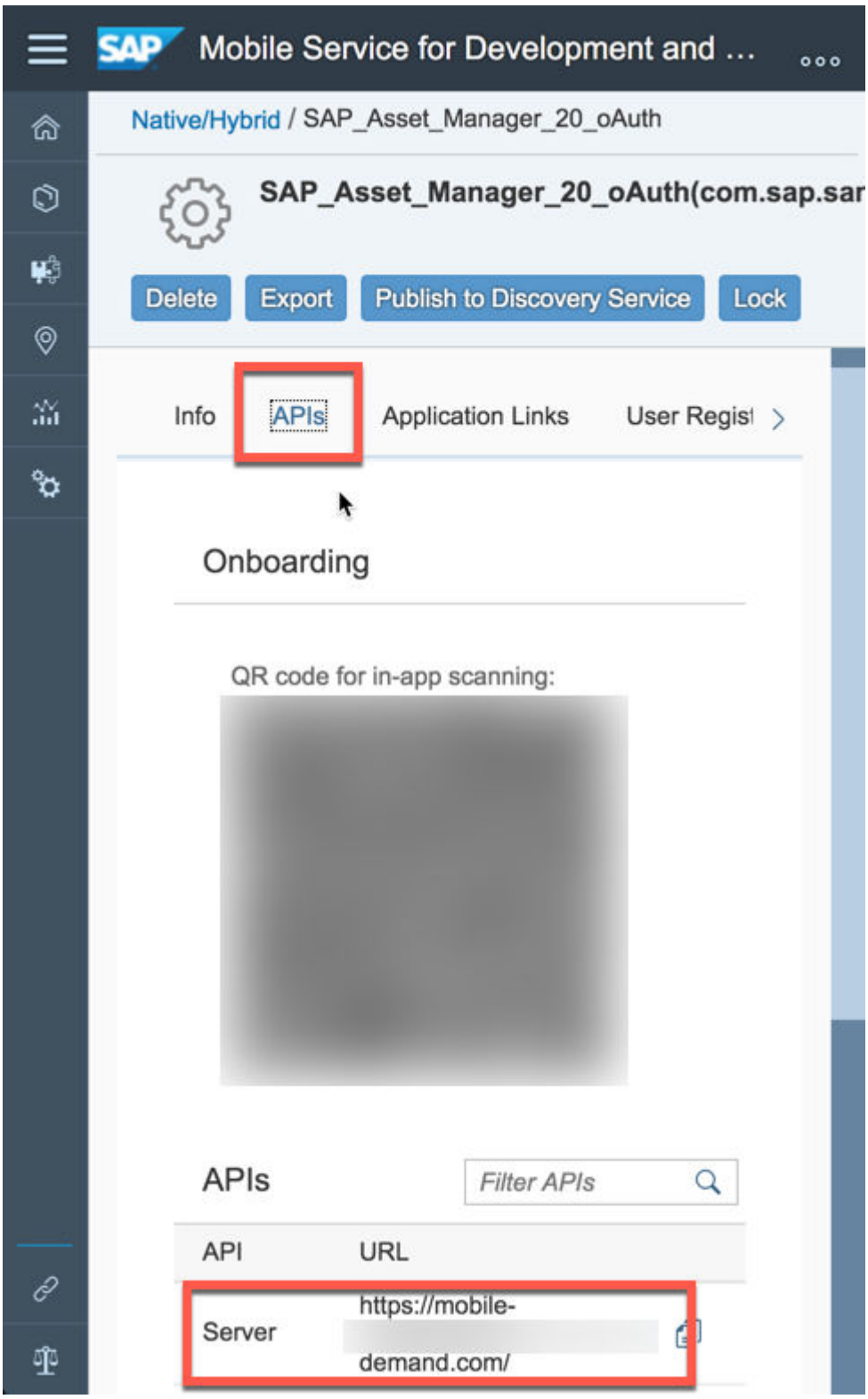
2. Create a folder to contain the installation files (`SAPAssetManager4.0Central`).
3. Extract the SAP Asset Manager branding SDK:
   a. Unzip `ASSET MGR BRANDING SDK 1911`.
   b. Copy the `SAPAssetManagerCentral/SAM.mdkproject` folder to the `SAPAssetManager4.0Central` folder.
   c. Set up the SAP Asset Manager Mobile Development Kit project folders:
      1. In the `SAPAssetManager4.0Meter/SAM.mdkproject/metadata` folder, create a folder named `meter.definitions`.
      2. In the `SAPAssetManager4.0Central/SAM.mdkproject` folder, create a folder named `extensions`.
4. Extract the SAP Asset Manager metadata:
   a. Unzip the `ASSET MANAGER METADATA 1911` file.
   b. Copy all contents of the ZIP file to `SAPAssetManager4.0Central/SAM.mdkproject/metadata/sam.definitions`.
   c. If present, delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0Central/SAM.mdkproject/metadata/sam.definitions` folder.
5. Extract the Asset Central SAP Asset Manager metadata:
   a. Unzip the `ASSETMGR ASSETCENTRAL META 2.0` file.
   b. Copy all contents of the ZIP file to `SAPAssetManager4.0Central/SAM.mdkproject/metadata/central.definitions`.
   c. If present, delete the `SIGNATURE.SMF` file from the `SAPAssetManager4.0Central/SAM.mdkproject/metadata/central.definitions` folder.
6. Extract the MDK plug-in SDK in one of the following ways, based on whether you're building either an iOS or an Android client:

iOS Client	Android Client
1. **Unzip the iOS subfolder found in the `MDK PLUG-IN SDK 4.0` zip file.** 2. **Choose your architecture from one of the following folders:** ○ **Release-iphoneos** ○ **Release-iphonesimulator** ○ **Release-fat (contains both the iphoneos and the iphonesimulator architectures)** 3. **Copy the following folders from your selected architecture folder to the `SAPAssetManager4.0Central/SAM.mdkproject/extensions` folder:** ○ `extension-Analytics` ○ `extension-BarcodeScanner` ○ `extension-FieldDataCapture`	1. Unzip the Android subfolder found in the `MDK PLUG-IN SDK 4.0` zip file. 2. Copy the following folders from your `Universal` folder to the `SAPAssetManager4.0Central/SAM.mdkproject/extensions` folder: ○ `extension-Analytics` ○ `extension-BarcodeScanner` ○ `extension-FieldDataCapture` ○ `extension-MapFramework` ○ `extension-HierarchyFramework`

iOS Client	Android Client
○ `extension-MapFramework` ○ `extension-HierarchyFramework`	

7. Configure the connection to SAP Cloud Platform mobile services:
   a. Retrieve the following information to establish a connection between the SAP Asset Manager application and the SAP Cloud Platform mobile services:
      - **AppId:** Note the ID under the Mobile Development Kit that you created in *Step 6* of the Creating an Application in SAP Cloud Platform mobile services procedure.
      - **ClientID:** Note the oAuth client ID that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

- **SapCloudPlatformEndpoint:** Find the Endpoint setting inside the application list of APIs on the *Mobile Services* under the *Server* API:



> **Note**
>
> By default, the *Server API* has a / at the end of the endpoint URL. Do not add this / into your connection settings.

- **AuthorizationEndpointURL:** Note the oAuth authorization endpoint URL that you created in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **RedirectURL:** Note the callback URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.
- **TokenURL:** Note the token URL that is automatically generated with the creation of the oAuth client in *Step 14* of the Creating an Application in SAP Cloud Platform mobile services procedure.

b. Choose your client configuration:

To preconfigure your client to connect to your mobile application, add the information retrieved in the above step to the *ConnectionSettings* block. When adding additional entries, include a comma after the existing *EnableOverrides* entry. An example is included in the table for reference when you're adding entries to your file.

If you want to use SAP Asset Manager against different back-end mobile applications (ex: DEV and QA), leave the *ConnectionSettings* as is, and build an onboarding URL for users using the values found in *Step 7a*. See the example in the table for further information on how to connect a client using either of the methods.

Preconfigured Client	Onboarding URL
You can generate a sample onboarding link to overwrite these values on a device with the following format in a URI:	You can generate an onboarding link to overwrite the values discussed in this substep on a device. Use the following format in a URI:
Sample Code … "DetailLabelViewText": "$(L,detail_label_view_text)", "ConnectionSettings": { "EnableOverrides": true, "AppId": "<Insert AppID value here>", "ClientId": "<Insert ClientID value here>", "SapCloudPlatformEndpoint": "https://<Insert cloud platform endpoint URL here>", "AuthorizationEndpointUrl": "https://<insert authorization endpoint URL here>", "RedirectUrl": "https://<insert redirect URL here>", "TokenUrl": "https://<insert token URL here>" },	**Sample Code** samclient://?AppId=<Insert AppID value here> &ClientId=<Insert ClientID value here> &SapCloudPlatformEndpoint=<Insert cloud platform endpoint URL here> &AuthorizationEndpointUrl=<insert authorization endpoint URL here> &RedirectUrl=<insert redirect URL here> &TokenUrl=<insert token URL here> &ServiceTimeZoneAbbreviation=<insert timezone abbreviation here>
Using the example as a guide, insert your own connection-specific values where they belong.	Save any changes you make.
Save any changes you make.	

8. Edit the project settings:
    a. Open the `SAPAssetManager4.0Central/SAM.mdkproject/MDKProject.json` file in a text editor. Edit app information such as:
        - Application name on the home screen
        - App version
        - Bundle ID to uniquely identify the application on the device
        - URL scheme for onboarding URLs
    b. Save any changes you make.
9. Set up the Mobile Development Kit Client SDK:
    a. Unzip the Mobile Development Kit SDK associated with the version you're building.
        1. Run the Mobile Development Kit dependencies installer and confirm that your system is ready.
        2. Unzip `MDKClient_SDK.zip` to the new `SAPAssetManager4.0Central` folder.
    b. To install the necessary dependencies, open a Terminal prompt in the `SAPAssetManager4.0Meter/MDKClient_SDK` directory. Run either `./install.command` on a Mac or `./install.cmd` on a Windows PC. Note that you can build both iOS and an Android on a Mac. You can only build Android if you're using a Windows PC.

> **i Note**
>
> An internet connection is required. If you're connecting to the internet through a proxy, configure your settings before running the `./install.command` or `./install.cmd` command.

10. Create the SAP Asset Manager client:
    a. Open a Terminal prompt in the `SAPAssetManager4.0Central/MDKClient_SDK` directory.
    b. Run the `create-client.command` command if you are on a Mac. Run the `create-client.cmd` command if you are on Windows.
    c. You can either specify command-line arguments to point to the `SAM.mdkproject` and the type of client (either device or simulator) you are building, or the script prompts you for the information.

> **Sample Code**
>
> on a Mac

```
$ ./create-client.command
? Enter the path of the .mdkproject directory. ../SAM.mdkproject
Using ../SAM.mdkproject
Using /Users/.../sdk for out directory
? Would you like to build for iOS or Android or All? ios
Building client for ios
? Would you like to build for device or simulator of iOS? device
Building client for device of iOS
Creating application SAPAssetManager4.0Central
```

> **Sample Code**
>
> on a Windows PC

```
>create-client.cmd
? Enter the path of the .mdkproject directory. ..\SAM.mdkproject
Using ..\SAM.mdkproject
Using C:\...\mdk for out directory
Building client for Android
```

```
Creating application SAPAssetManager4.0Central
```

d. Remove `demo.js` from the `CentralAssetManager/app` directory.

Results

After `create-client.command` for iOS or `create-client.cmd` for Windows finishes, you're ready to run the client either on the mobile device or on a simulator.

Next Steps

Open a Terminal prompt in the resulting client directory (ex: `SAPAssetManager4.0Central/MDKClient_SDK/MeterAssetManager`). Run either the `tns run ios` or the `tns run android` command to start the application, based on your platform.

For iOS installations only, continue to the Allowing Custom URI Schemes [page 9] procedure.

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 a 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.

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.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

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.

www.sap.com/contactsap

THE BEST RUN SAP