IBM Maximo Anywhere
Version 7 Release 6

Configuring Maximo Anywhere mobile apps

Note

Before using this information and the product it supports, read the information in "Notices" on page 31.

This edition applies to version 7, release 6, modification 2 of IBM® Maxime Anywhere and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Customizing Maximo Anywhere mobile apps

You can modify the user interface of Maximo Anywhere mobile apps, such as changing field labels, adding fields, or adding views by updating the application artifact files for the app in IBM MobileFirst® Platform Studio.

Getting started

Before you modify the mobile apps, set up the development environment by installing the Worklight Studio on the build server.

MobileFirst Studio development environment

MobileFirst Studio is an Eclipse-based integrated development environment that you can deploy with Maximo Anywhere. You use MobileFirst Studio to configure and test the Maximo Anywhere apps and to manage the build and deployment process.

You configure the mobile app by editing the application definition, the `app.xml` file, in the XML editor in MobileFirst Studio. By using the Design tab in the XML editor, you can easily find the elements in the file and change or add to the content. When you save your changes, the app is built automatically and you can test your changes. You can then deploy the updated apps to the server.

Each element in the application definition requires an ID. When you have filled in the **ID** field, this customization of the application definition is preserved during any upgrades.

Before you deploy the mobile apps to the production environment, you can preview and test all updates to mobile apps in the mobile browser simulator. You must select Google Chrome as your external web browser to preview in the mobile browser simulator.

To preview your changes, right-click on the application folder and select **Run As** > **Preview**.

Related information:

MobileFirst Studio overview

Preparing the Maximo Anywhere environment

Before you can build and deploy the Maximo Anywhere apps, you must install development tools that are specific to the mobile platform.

About this task

To build the mobile apps for iOS devices, a Mac OS X computer is required. The order of installation is different on Mac OS X computers. For more information, see the technote Installing a Maximo Anywhere development environment on Mac OS X.

Procedure

1. Prepare the computer for building the mobile apps:

Option	Description
Android	Install the Android development tools.
iOS	Install the iOS development tools.
Windows	Install the Windows development tools.

2. Install Maximo Anywhere.
3. Create a MobileFirst runtime environment.
4. Deploy the apps from the command line.
5. Optional: Install an integrated development environment.

Installing the Android development tools

Oracle JDK and Android SDK are required to build Android mobile apps.

Procedure

1. To install Oracle JDK version 8.0, from the Oracle Java™ SE downloads page, download the Java SE Development Kit for your operating system.
2. Run the executable file and proceed through the installation program.
3. Set the *JAVA_HOME* environment variable and specify the path to the directory where the JDK was installed:

Option	Description

On Windows	1. Go to your computer's Advanced System Settings.
	2. On the Advanced tab, click **Environment Variables**.
	3. Under System Variables, click **New**.
	4. Specify `JAVA_HOME` as the variable name and enter the path to the JDK installation directory as the variable value.
	Environment variable values cannot be separated by a space. If the Java installation directory contains a space in the path name, specify the shortened path name. For example, on Windows operating systems, enter a path such as `C:\Progra~1\Java\jdk1.8.0_x`.
	5. Close all open windows.
	6. Verify that the *JAVA_HOME* system environment variable is defined by opening a new command prompt and running the following command:
	`SET JAVA_HOME`
	A value is returned such as:
	`JAVA_HOME=C:\Progra~1\Java\jdk1.8.0_x`

Option	Description

On UNIX or Linux	1. At a command line, run the following command:
	`vi ~/.bash_profile`
	2. Set the variables by running the following command and replacing the *java_path* variable with the Java path that you specified:
	`export JAVA_HOME=`*`java_path`*
	`export PATH=$JAVA_HOME/bin:$PATH`
	3. Save and close the `.bash` profile, and run the following command to apply the changes:
	`source ~/.bash_profile`
	4. Verify that the *JAVA_HOME* system environment variable is defined by opening a new command line and running the following command:
	`echo $JAVA_HOME`

4. To install the Android SDK, go to the Android developers' website and click to see other download options.
5. In the section with the SDK tools, download the SDK for your operating system and extract the compressed file on your computer.
6. Start the installation wizard.

Option	Description
On Windows	From the extracted directory, run the `SDK Manager.exe` file.
On UNIX or Linux	Open a terminal and navigate to the *`android_home`*`/tools/` directory, then run the following command: `android sdk`

7. Under Tools, select Android SDK Tools Revision 22 or later and the latest Android SDK build tools. Also, select an Android package with an API of 14 or greater and click **Install Packages**.

Installing the iOS development tools

A Mac OS X computer with Oracle JDK and Xcode installed is required to build the iOS mobile apps.

About this task

You use your Apple ID to enroll in a developer program. You can enroll in the iOS Developer Program as an individual or a company in which an individual is a one-person team.

You can also enroll in the iOS Developer Enterprise Program as a company, which authorizes you to create proprietary in-house iOS apps. The procedure for setting up the iOS Developer Enterprise account might include extra steps.

The person who creates the team becomes the *team agent* who is the legal contact and administrator of the team and has all privileges and full access to Member Center and iTunes Connect. The team agent is required to complete steps 5-7. iOS developers might require the assistance of the team agent to complete the procedure.

You create provisioning profiles to manage the use of iOS apps that you develop.

A *development provisioning profile* regulates the development and testing of apps to a specific number of devices. Development provisioning profiles for the apps must contain the IDs of the devices on which the app is installed. You can register more iOS devices and add them to the provisioning profiles.

A *distribution provisioning profile* provides access to the app store from all of the devices in your enterprise.

Procedure

1. Install Oracle JDK version 8.0.
   a. From the Oracle Java SE downloads page, download the Java SE Development Kit for your operating system.
   b. Run the executable file and complete the installation program.
   c. In a terminal, go to the home directory `cd ~`.
   d. Enter `echo $JAVA_HOME`.
   e. If the result is empty, enter `sudo nano .bash_profile`. This step requires your system password.
   f. In the .bash_profile file window, enter `export JAVA_HOME=$(/usr/libexec/java_home)`, and save and close the file.
   g. To reload the terminal and read what you put in the files, enter `source ~/.bash_profile`.
   h. Enter `echo $JAVA_HOME`. Verify that the path is `/Library/Java/JavaVirtualMachines/`*`jdk_version`*`/Contents/Home`.
2. Create an `environment.plist` file in the `~/Library/LaunchAgents/` directory and specify the following content, which substitutes your actual level of JDK. Replace the value *jdk1.8.0_x.jdk* with the value of your installed version.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>my.startup</string>
  <key>ProgramArguments</key>
  <array>
    <string>sh</string>
    <string>-c</string>
    <string>
    launchctl setenv JAVA_HOME /Library/Java/JavaVirtualMachines
    /jdk1.8.0_x.jdk/Contents/Home/
    </string>
  </array>
  <key>RunAtLoad</key>
  <true/>
</dict>
</plist>
```

The `.plist` file activates after you reboot the system. You can also use the command `launchctl load ⌂/Library/LaunchAgents/environment.plist` to launch it immediately.

3. Create an Apple ID by registering as an Apple developer in the Apple Registration Center.
4. Download and install Xcode IDE, including the iOS SDK and Simulator, from the Mac App Store.
5. Add your Apple ID to Xcode.
6. If you are the team agent, in Xcode create the certificate for the distribution provisioning profile. Download the certificate to your local environment.
7. Register the App IDs.

   Create a unique ID for each app that you support. The app ID that you specify must be unique: Apple does not allow duplicate app IDs.

   The following table contains examples of bundle IDs that can be specified for the Maximo Anywhere apps. Replace the *company_name* value with your company name.

App name	Example bundle ID
Asset Audit	com.*company_name*.maximoanywhere.AssetAudit
Asset Data Manager	com.*company_name*.maximoanywhere.AssetDataManager
Inspection	com.*company_name*.maximoanywhere.Inspection
Issues and Returns	com.*company_name*.maximoanywhere.IssuesReturns
Physical Count	com.*company_name*.maximoanywhere.PhysicalCount
Service Request	com.*company_name*.maximoanywhere.ServiceRequest
Transfers and Receiving	com.*company_name*.maximoanywhere.Transfers
Work Approval	com.*company_name*.maximoanywhere.WorkApproval
Work Execution	com.*company_name*.maximoanywhere.WorkExecution

   To specify an identifier to represent a single app, click **Explicit App ID** and enter a unique ID for the app. Repeat this step for each app that you support.

8. From the `*maximoanywhere_home*\MaximoAnywhere\apps\*app_name*` directory, open the `application_descriptor.xml` file and change the bundle ID strings to match the bundle IDs that you created.
9. To support command line builds of our app, you must create a distribution provisioning profile. Ensure that you select the distribution certificate when creating this provisioning profile.
   a. Create a distribution provisioning profile for each app by using Member Center. When you generate a distribution provisioning profile, you are not required to provide IDs of the devices.
   b. Download each app's provisioning profile to your Mac OS X computer. Every time that a provisioning profile is updated, you must download the profile to the build server.

   When you run the app build and deployment process, the provisioning profiles are collected and stored in the iOS application archive file (IPA).

10. Run the following command:

```
./build.sh all
```

11. Optional: To test the app on a local device using Xcode, you must register the Device IDs for all of your test devices. You must also create a Developer Provisioning Profile for the app. While creating the provisioning profile,

ensure that you select your developer certificate while creating the app and also the Device IDs of your test devices.

a. Create a developer certificate for a development provisioning profile. Download the certificate and add it to the Keychain application.
b. Register the Device IDs in Member Center. You can locate the unique device identifier (UDID) by connecting your device to the Mac OS X computer while Xcode or iTunes are running.
c. Download the developer provisioning profile to your Mac OS X computer.

Installing an integrated development environment

You can create an integrated development environment (IDE) by installing the Eclipse IDE and MobileFirst Studio. You can use the IDE to develop, test, and configure mobile apps before you deploy them to MobileFirst Server or to mobile devices.

Before you begin

- Ensure that Maximo Anywhere is installed on the computer where you are installing the IDE.
- It is recommended that you install the IDE on a computer that is independent of the server.
- Depending on the mobile platform that you plan to develop apps for, ensure that you installed the Android, iOS, or Windows development tools.
- Oracle JDK version 8 (v1.8) is required to run Eclipse and to build apps for Android, iOS, and Windows devices.
- To preview the apps in the MobileFirst Studio mobile browser simulator, install Google Chrome.
- To allow Eclipse to install the required plugins, ensure that your firewall is temporarily disabled.
- If you are installing an integrated development environment on a Mac OS X computer, follow the instructions in this technote: Installing a Maximo Anywhere development environment on Mac OS X.

About this task

MobileFirst Studio is an Eclipse-based development environment that can be used to configure the Maximo Anywhere apps. You install MobileFirst Studio from the Eclipse integrated development environment (IDE) workbench. A set of platform development tools are also required to build and deploy the mobile apps.

You can use the mobile browser simulator, the Android Emulator, or the iOS Simulator to view and test the mobile apps.

You can customize the mobile apps when they are deployed. For more information, see Customizing Maximo Anywhere mobile apps.

Procedure

1. Install Eclipse IDE:
   a. Go to Eclipse IDE for Java EE Developers and download Eclipse IDE for Java EE Developers (Eclipse Kepler 2).
   b. Extract the compressed folder and run the Eclipse application.
2. Set up your Eclipse development environment:

   a. Check whether the Eclipse JSDT plug-in is installed. The JavaScript Development Tool (JSDT) plug-in helps you navigate through the Java Script code in Eclipse.
      1) Select **Help** > **About Eclipse** > **Installation Details** and click the **WTP** icon (Eclipse Web Tools Platform).
      2) In the About Eclipse Features window, browse the Feature Name list for Eclipse JavaScript Development Tools.
   b. If the JavaScript Development Tools plug-in is not already installed, complete the following steps:
      1) Select **Help** > **Install New Software** > **Add**.
      2) In the Add Repository window, specify the following URL in the **Location** field: `http://download.eclipse.org/webtools/repository/kepler`.
      3) In the **Name** field, specify `Eclipse WTP`.
      4) Select **Web Tools Platform (WTP) 3.5.1 (or later)** > **JavaScript Development Tools** and install the plug-in.
   c. Set Google Chrome as the default web browser by selecting **Window** > **Preferences** > **General** > **Web Browser** > **Use external web browser**. If **Chrome** is not available in the External web browsers window, click **New**, specify `Chrome` in the **Name** field and browse to the `Chrome.exe` file which is in the `C:\Program Files (x86)\Google\Chrome\Application` directory on Windows systems. Click **OK** to apply your changes. Alternatively, you can set your default web browser to Chrome.

3. Install MobileFirst Studio Consumer Edition:
   a. In Eclipse, select **Help** > **Install New Software** > **Add**.
   b. In the Add Repository window, click **Archive**.
   c. Browse to the `MaxAny_762_MobileFirst_71/MobileFirstStudio.zip` folder and click **Open** > **OK**.
   d. On the Available Software pane, select **IBM MobileFirst Platform Studio Development Tools**. The following items are selected for installation: IBM Dojo Mobile Tools, IBM jQuery Mobile Tools, and IBM MobileFirst Platform Studio.
   e. Click **Next** > **Next** > **Finish**.
   f. Restart Eclipse to apply the changes.
4. Import the `MaximoAnywhere` project into MobileFirst Studio.
   a. In Eclipse, click **File** > **Import** > **General**, select **Existing Projects into Workspace** and click **Next**.
   b. Click **Select root directory** and specify the path to `\IBM\Anywhere`.
   c. Select the `MaximoAnywhere` project and click **Copy projects into workspace**. Click **Finish**.
   d. In the Project Explorer pane, expand all nodes. Right-click the `OSLCGenericAdapter` folder and select **Run As** > **Deploy MobileFirst Adapter**.
   e. Right-click the application folder and click **Run As** > **Run on MobileFirst Development Server**.
   f. To preview the app, right-click the application folder and click **Run As** > **Preview**.
5. To build apps for Android devices, install the Android Development Tools (ADT) plug-in:
   a. In Eclipse, select **Help** > **Install New Software** > **Add**.

b. In the Add Repository window, specify the following URL in the **Location** field: `https://dl-ssl.google.com/android/eclipse/`.
c. In the **Name** field, specify `Android Development Tools` and click **OK**.
d. On the Available Software pane, click **Select All** > **Next** > **Next** > **Finish**.
e. Restart Eclipse and specify a workspace directory.
f. Open the Eclipse Preferences window and specify the Android SDK path. Click **Apply** > **OK** to apply the changes.
g. On the Welcome to Android Development pane, select **Use existing SDKs** and browse to the SDK directory. Click **OK** > **Next** > **Finish**.

Deploying apps by using MobileFirst Studio

MobileFirst Studio includes an embedded instance of MobileFirst Server, which means that you do not need to deploy the WAR file. MobileFirst Studio also includes the Mobile Browser Simulator for testing your apps.

Before you begin

- Set up Maximo Asset Management for Maximo Anywhere.
- If you plan to preview the deployed apps in the Mobile Browser Simulator, Google Chrome must be installed in the development environment. The preview function for deployed apps does not support other web browsers.

Procedure

1. In MobileFirst Studio, import the Maximo Anywhere project.
   a. Right-click inside the Project Explorer, and select **Import**. From the Import window, select **Existing Projects into Workspace** and click **Next**.
   b. Browse to the *`maximoanywhere_home`* folder, select the `Anywhere\MaximoAnywhere` folder, select **Copy projects into workspace**, and click **Finish**.
2. Set the Ant file path.
   a. From the **Window** menu, select **Preferences**. Expand **Ant** and then select **Runtime**.
   b. Select **Ant Home Entry** and click **Ant Home**.
   c. In the Browse for Folder window, expand ***maximoanywhere_home*** > **Anywhere** > **MaximoAnywhere** > **build** > **tools** > **ant** and click **OK**.
3. Add the `build.xml` file to the Ant view. Select the `build.xml` file from **MaximoAnywhere** and drag the `build.xml` file to the Ant view.
4. Expand the `build.xml` file in the Ant view and double-click the **all** task.
5. From the Project Explorer, deploy the adapter.
   a. Select **MaximoAnywhere > Adapters** and right-click **OSLCGenericAdapter**.
   b. Select **Run As > Deploy MobileFirst Adapter**.
6. Build and deploy the Maximo Anywhere Asset Audit app.
   a. Under **MaximoAnywhere > apps**, right-click **AssetAudit**.
   b. Select **Run As > Run on MobileFirst Development Server**.
7. Repeat step 6 for each of the following applications:
   - AssetDataManager
   - Inspection
   - IssuesReturns
   - PhysicalCount

- ServiceRequest
- Transfers
- WorkApproval
- WorkExecution

8. Verify the application deployment in MobileFirst Operations Console by right-clicking the app and selecting **Run As > Preview**.
9. Test your app on your mobile device, Android emulator, or iOS Simulator. To test the app on the Android emulator, you must first set up the emulator. To test the app on a mobile device, connect that device to your computer.

Device type	Steps
Android	1. In the Project Explorer, select the Android project that is under the app that you are testing and select **Run as > Android Application**. 2. Select **Launch a new Android Virtual Device** for the Android emulator or **Launch a new Android Device** for an Android mobile device.
iOS	1. Under **MaximoAnywhere > apps >** ***app_name*** **> iphone**, select **Run as > Xcode project**. 2. From the Xcode window, choose the simulated device or mobile device on which you want to test the app, and click the **Play** icon.
Windows	1. Open Microsoft Visual Studio. 2. From the menu bar, select **FILE** > **Open** > **Project/Solution**. 3. From the `MaximoAnywhere/apps/`*`application_name`*`/`*`windows`*`/native` directory, select the `*.jsproj` file. 4. In Visual Studio, select the `index.html` file, and then click **Simulator** to run the simulator.

Integrating Maximo Anywhere with Maximo Asset Management

Maximo Anywhere integrates with Maximo® Asset Management applications by using Open Services for Lifecycle Collaboration (OSLC).

OSLC resources

Maximo Anywhere provides OSLC resources that defines the metadata for business objects of OSLC service providers. The OSLC resources are used by the mobile apps to retrieve and process application data from the service providers in Maximo Asset Management.

The resource metadata includes Resource Description Framework (RDF) format files that are defined for each business object in the integration between Maximo Anywhere and Maximo Asset Management. The resources for Maximo Anywhere is in the `Anywhere\MaximoAnywhere\oslc-docs\resources\rdf\oslc` directory.

The OSLC resources that Maximo Anywhere provides includes predefined object structures, service providers, and RDFs. You can also define additional resource data in Maximo Asset Management for new resources that you want to use in the mobile apps.

After you define the resource data, you must import the RDF files to Maximo Anywhere so that the data can be used by the mobile apps. You implement the resources in the mobile apps by updating the application definition file and deploying a new application version.

Typically, you use the OSLC resources in the following scenarios.

Adding fields from Maximo Anywhere resources
When you add a field to the mobile app from the current Maximo Anywhere resources. If the object structure and RDF exists in Maximo Anywhere but is not included in the app, you update the `app.xml` file.

Adding new fields from Maximo Asset Management to Maximo Anywhere
When you create a business object in Maximo Asset Management for a new field, the object structures by default retrieve all fields that are defined by the business object. To use the new field in Maximo Anywhere, you must import the RDF from Maximo Asset Management to Maximo Anywhere and then update the `app.xml` file.

Adding a child object to an existing object structure
When you add a child object in Maximo Asset Management to an object structure that exists in Maximo Anywhere. In Maximo Asset Management, you update the resource that has the object structure to include the child object. To use the child object in Maximo Anywhere, you must import the RDF from Maximo Asset Management to Maximo Anywhere and then update the `app.xml` file.

Creating an object structure for an existing object
When you create an object structure for an existing object, such as a domain or value list, in Maximo Asset Management and you want to add the object structure to the field. In Maximo Asset Management, you create the object structure and update the resource with the object structure. To use the domain in Maximo Anywhere, you must import the RDF from Maximo Asset Management to Maximo Anywhere and then update the `app.xml` file.

The following figure shows how object data can be configured as resource data and used in Maximo Anywhere.

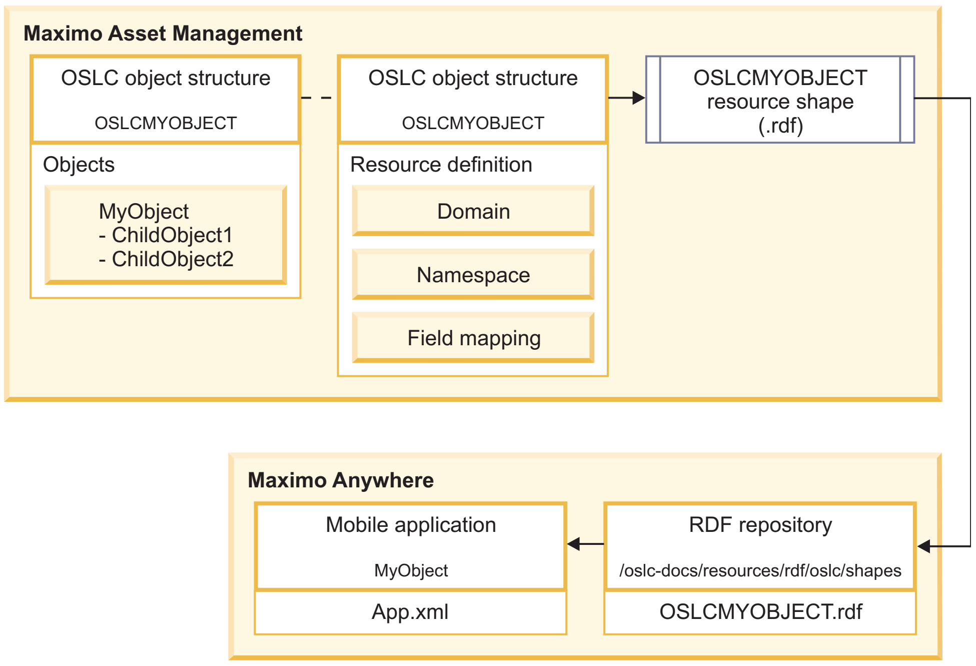

Figure 1. How object data can be configured as resource data

Creating OSLC resources in Maximo Asset Management

OSLC resources are needed for retrieving and processing data in mobile apps. The OSLC resources that you define must be based on the associated object structure in Maximo. To define additional resource data for new resources that you want to use in the mobile apps, you create the object structure and the resource in Maximo Asset Management.

About this task

OSLC resources are implemented by object structures. An *object structure* is the common data layer that the integration framework uses for processing data. When a resource is created as an OSLC resource, Maximo Anywhere can make requests to query, create, update, or delete the resource data.

Procedure

1. In the Object Structures application, click **New Object Structure** and specify an object structure identifier.
2. In the **Consumed By** field, specify **OSLC** as the module that uses the object structure.
3. Specify the source objects for the object structure and save the record.
4. In the OSLC Resources application, create a resource.
5. Specify the object structure that you created in the Object Structures application.
6. Specify the domain name and the default namespace URI. You can specify an existing domain and namespaces for the resource, for example `SmarterPhysicalInfrastructure` and `http://jazz.net/ns/ism/asset/smarter_physical_infrastructure#`. Or you can define a new domain and namespace by using the **Select Action** menu.

7. Save the record. The resource shape document is updated to include the object structure for the field.
8. Optional: Verify that the resource shape document is updated at the following URL `http://`*hostname*`/maximo/oslc/shapes/`*object_structure*.

What to do next

Import OSLC resources into Maximo Anywhere.

Importing OSLC resources to Maximo Anywhere

After you create OSLC resources in Maximo Asset Management, you need to import the updated metadata to Maximo Anywhere.

Before you begin

Verify that the address properties for the service provider are correct in the `build.properties` file. The `build.properties` file is in the `Anywhere\MaximoAnywhere` folder.

- `adapter.name=`*`service_provider_name`*
- `adapter.connection.protocol=http`
- `adapter.connection.domain=`*`ip`*
- `adapter.connection.port=`*`port_number`*
- `adapter.connection.context=`*`context`*

About this task

You run an import utility Ant script to import the resource data from Maximo Asset Management to the Resource Description Framework (RDF) files in Maximo Anywhere.

The import utility is named `anywhere-rdfs-puller.xml` and is in the `Anywhere\MaximoAnywhere` folder. The utility includes all the shape documents that are used by the service provider in Maximo Asset Management for the Maximo Anywhere mobile apps.

Procedure

1. Edit the `anywhere-rdfs-puller.xml` file and add the appropriate RDFs. For example, if you created the resource for myRelatedObject, you add the URI `<downloadOneRdf context="/oslc/shapes/oslcwodetail/myRelatedObject" />` to the `anywhere-rdfs-puller.xml` file.
2. To run the import utility in a development environment, specify the user name and password as arguments for the file.
   a. In MobileFirst Studio, add the `anywhere-rdfs-puller.xml` Ant file to the Ant view.
   b. Right-click on the `anywhere-rdfs-puller.xml` Ant file and select **Run as** > **Ant Build**.
   c. On the Main tab, add the user name and password for the adapter in the **Arguments** field.`-Dadapter.connection.user=<`*`user_name`*`>` `-Dadapter.connection.password=<`*`password`*`>`
3. Import the OSLC resource data.

Environment	Action
In a development environment in MobileFirst Studio	Run the `anywhere-rdfs-puller.xml` file from the Ant view.
In a production environment	Run this command: `ant -f anywhere-rdfs-puller.xml -Dadapter.connection.user=<user_name> -Dadapter.connection.password=<password>`

4. Verify that the RDF is imported and that the RDF content, name, and attributes are correct in the `Anywhere\MaximoAnywhere\oslc-docs\resources\rdf\oslc` directory.

What to do next

To implement the resources in the mobile apps, update the application definition and deploy a new application version.

Creating and modifying fields in mobile apps

To develop a Maximo Anywhere mobile app, you can add fields to the mobile apps, rename fields and views, and also add values list to fields.

Changing field labels

The labels for fields in the mobile apps are defined in the `app.xml` file. You can change any of the labels to match your implementation of Maximo Anywhere.

Procedure

1. In MobileFirst Studio, go to the `Anywhere\MaximoAnywhere\apps\`*`app_name`*`\artifact` directory and open the `app.xml` file with the XML editor.
2. On the Design tab, expand the view that contains the label attribute you want to change and enter the new name.
3. If you translate the mobile app, change the name of the label in the `artifact.js` file for each supported language. For example, if the base language of the mobile app is English and you translate the app to French, change the label name in the `artifact.js` file in the `Anywhere\MaximoAnywhere\apps\WorkExecution\common\js\application\translation\nls\fr` directory.
4. Save your changes. The application is built automatically.
5. Optional: To preview your changes in the mobile browser simulator, right-click on the application folder and select **Run As** > **Preview**.

What to do next

Deploy the app to the server

Rearranging fields in a mobile app

The order of how fields appear in a mobile container view is defined in the `app.xml` file. You can change the position of the fields for users to enter the information in an order that suits business requirements.

About this task

Fields in a container view are defined as attributes within <groupitem> elements in the application definition file and are contained within a <group> element. The <group> element displays the fields on an app view. The order of attributes in a group reflects the order that the attributes appear on the screen.

A container view in a mobile app displays predefined fields and details of a record. For example, the Work Order Details view on the Work Execution app is a container view that displays the fields for the work order.

Procedure

1. In MobileFirst Studio, go to the `Anywhere\MaximoAnywhere\apps\`*`app_name`*`\artifact` directory and open the `app.xml` file with the XML editor.
2. Move the position of the field.
   a. On the Design tab, find and select the <group> element that contains the <groupitem> element that you want to move.
   b. On the Source tab, cut and paste the <groupitem> element to the new position.
3. Save your changes. The application is built automatically.
4. Optional: To preview your changes in the mobile browser simulator, right-click on the application folder and select **Run As** > **Preview**.

What to do next

Deploy the app to the server

Adding fields to mobile apps

The fields that you add to mobile apps are defined by OSLC Resource Description Framework (RDF) files. When you add fields to mobile apps, you specify the OSLC resource for the field as a resource attribute in the application definition file.

Before you begin

Run the RDF puller to import OSLC resources into Maximo Anywhere.

About this task

The OSLC resource must exist in Maximo Asset Management for the field that you are adding to the mobile app. You can use an existing OSLC resource for the field that you are adding. You can also create a field by creating the resource for the field in Maximo Asset Management and importing the resource RDF into Maximo Anywhere.

When you add a field to an app view, you must first define the resource for the field in the Data section of the application definition file. You then add the resource to the view in the UI section so that the field is shown on the app.

If you translate the mobile app, update the `artifact.js` file for each supported language to include the name of the field. The `artifact.js` files for each language are in the `Anywhere\MaximoAnywhere\apps\`*`app_name`*`\common\js\application\translation` directory.

Procedure

1. In MobileFirst Studio, go to the `Anywhere\MaximoAnywhere\apps\`*`app_name`*`\artifact` directory and open the `app.xml` file with the XML editor.
2. In the Data section, add the attribute to the resource that you want to add the field to.
   a. On the Design tab, find the <resource> element that you want to add the field to.
   b. On the Source tab, add the attribute that includes the shape document details to the <resource> element.

   For example, to add a field named Risk to the Work Order Details view in the Work Execution app, add the details of the risk attribute to the resource named workOrder:

```
<resource providedBy="/oslc/sp/WorkManagement"
   describedBy="http://jazz.net/ns/ism/work/smarter_physical_infrastructure#WorkOrder"
   name="workOrder" pageSize="200" class="application.business.WorkOrderObject>
 <attributes>
  <.....>
  <attribute name="risk" describedByProperty="spi_wm:risk" index="false" />
```

3. In the UI section, add the resource attribute to the view that you want to add the field to.
   a. On the Design tab, find the <view> element that you want to update.
   b. On the Source tab, add the resource attribute to the <group item> element within the view you are updating.

   For example, add the resource attribute for the **Risk** field to the view for Work Order Details:

```
<view id="WorkExecution.WorkDetailView" label="Work Order Details">
...
 <groupitem transitionTo="WorkExecution.DescriptionView">
  <text resourceAttribute="description" label="Description"
   editable="true" placeHolder="Tap to enter" />
 </groupitem>
 <groupitem>
  <text resourceAttribute="risk" label="Risk"
   editable="true" placeHolder="Tap to enter" />
 </groupitem>
```

4. Optional: Make the field editable by adding the value of `true` to the editable attribute and adding the value of `tap to enter` to the placeholder attribute. The placeholder attribute indicates that the field is editable and that you can enter any value in the field. To change the field to read only, change the value of the editable attribute to `false` and delete the placeholder attribute.
5. Save your changes. The application is built automatically.
6. Optional: To preview your changes in a mobile simulator, right-click on the application folder and select **Run As** > **Preview**.

What to do next

Deploy the app to the server

Related tasks:

"Creating OSLC resources in Maximo Asset Management" on page 11
OSLC resources are needed for retrieving and processing data in mobile apps. The OSLC resources that you define must be based on the associated object structure in Maximo. To define additional resource data for new resources that you want to use in the mobile apps, you create the object structure and the resource in Maximo Asset Management.

Adding values lists as lookups to fields

Some fields in a mobile app have value lists, or domains, from which users can select a value. You add a value list as a lookup in a field in the `app.xml`file.

Before you begin

Add the field that you want to associate the lookup of values with to the mobile app.

Run the Resource Description Framework (RDF) puller to import Open Services for Lifecycle Collaboration (OSLC) resources into Maximo Anywhere.

Maximo Anywhere includes OSLC resources for synonym domains and alphanumeric (ALN) domains. Verify that the OSLC resource is provided in Maximo Anywhere for the type of domain that you are adding. You can also create resources for a domain in Maximo Asset Management and import them into Maximo Anywhere.

About this task

To add a value list to a field, you add the value list as a resource and create a lookup for that resource. You can then add the resource and the lookup to the view for the field.

Procedure

1. In MobileFirst Studio, open the `app.xml` file, in an XML editor, for the mobile app that you want to update. The `app.xml` file is in the `Anywhere\MaximoAnywhere\apps\`*`app_name`*`\artifact` folder.
2. In the Data Definition section, add the resource data for the value list.

   For example, to add a value list to the field named Risk on the Work Order Details view of the Work Execution app, add the resource data for the domain named RISKDOMAIN to the data section of the `app.xml` file. RISK_DOMAIN is an ALN domain and has the values of high, medium, and low.

```
<!-- RISK DOMAIN -->
<resource providedBy="/oslc/sp/SmarterPhysicalInfrastructure"
describedBy="http://jazz.net/ns/ism/asset/smarter_physical_infrastructure#AlphaNumericDomain"
 name="riskDomain" pageSize="10" additionalData="true">
 <attributes>
  <attribute name="value" describedByProperty="spi:value" />
  <attribute name="domainid" describedByProperty="spi:domainid"/>
  <attribute name="alndomainid" describedByProperty="spi:alndomainid"/>
  <attribute name="description" describedByProperty="spi:description" />
 </attributes>
 <queryBases>
  <queryBase name="getRisk" queryUri="/oslc/os/oslcalndomain" />
 </queryBases>
 <whereClause clause="spi:domainid='RISKDOMAIN'" />
</resource>
```

3. In the UI section, create the lookup data for the value list and associate the resource with the lookup.

   For example, add the lookup for the RISK DOMAIN resource.

```
<!-- Risk Domain Lookup -->
<lookup id="WorkExecution.RiskLookup" label="Select Risk"
  resource="riskDomain" >
 <requiredResources>
  <requiredResource name="riskDomain"/>
 </requiredResources>
 <list resource="riskDomain">
  <listItemTemplate layout="Item1Desc1">
    <listtext resourceAttribute="value" layoutInsertAt="item1"
```

```
      cssClass="bold textappearance-medium"/>
     <listtext resourceAttribute="description" layoutInsertAt="desc1"
       cssClass="bold textappearance-medium"/>
   </listItemTemplate>
  </list>
  <returnAttributes>
   <returnAttribute sourceAttribute="value" targetAttribute="risk" />
  </returnAttributes>
 </lookup>
```

4. In the UI section, associate the new lookup with the field.
   a. On the Design tab, find the <view> element for the field that you want to add the lookup to.
   b. On the Source tab, add the lookup to the <group item> element within the view that you are updating.

   For example, associate the risk domain lookup with the Work Order Details view.

```
<view id="WorkExecution.WorkDetailView" label="Work Order Details"
 resource="workOrder" >
 <...>
 <container resource="workOrder">
 <...>
  <groupitem>
   <text resourceAttribute="risk" label="Risk"
     editable="true" lookup="WorkExecution.RiskLookup"
     lookupAttribute="risk" placeHolder="Tap to enter" />
  </groupitem>
 <...>
</view>
```

5. Save your changes. The application builds automatically.
6. Optional: To preview and test your changes in a mobile simulator, right-click on the application folder and select **Run As** > **Preview**.

What to do next

Deploy the app to the server

Related tasks:

"Creating OSLC resources in Maximo Asset Management" on page 11
OSLC resources are needed for retrieving and processing data in mobile apps. The OSLC resources that you define must be based on the associated object structure in Maximo. To define additional resource data for new resources that you want to use in the mobile apps, you create the object structure and the resource in Maximo Asset Management.

Related information:

Predefined OSLC resources and object structures

Adding related records to mobile apps

Objects that you can add as related records to mobile apps are defined by OSLC Resource Description Framework (RDF) files. In Maximo Asset Management, you can add a related object to an OSLC resource and use the metadata for the object as a related record in a Maximo Anywhere app.

Before you begin

Run the RDF puller to import OSLC resources into Maximo Anywhere.

About this task

When you add a related or child object in Maximo Asset Management to an OSLC object structure that exists in Maximo Anywhere, you update the parent OSLC resource to include the child object. To use the child object as a related record in a mobile app, you import the RDF from Maximo Asset Management to Maximo Anywhere. After you import the RDF, you define the resource and the views for the related record in the app artifact files.

The resource for the child object must exist in the `app.xml` file so that the fields can appear in a mobile app. To show the related record, you update the `app.xml` file to add a view to show the details of the record and a view to show the list of records that are related to the mobile app.

For example, the communication log is a child object of the Work Order Maximo business object (MBO). A list of communication logs can be related to the Work Order. To use communication logs as related records on the Work Order Details view in the Work Execution app, you add the object for the communication log to the work order OSLC object structure and associated OSLC resource in Maximo Asset Management. You import the updated metadata into Maximo Anywhere and add the attribute for the communication log list to the `app.xml` file so that the records for the communication logs and their fields can appear the Work Order Details view.

Procedure

1. In Maximo Asset Management, add the object for the related record to the OSLC resource.
   a. In the Object Structures application, open the object structure record that is associated with the resource that you want to add the object to.
   b. Add the child object as a source object.
   c. Specify the parent object and the relationship and save the record. The OSLC resource that is associated with the object structure is automatically updated to include the object.
   d. In the OSLC Resources application, find the OSLC resource and verify that the child object is added.

   For example, to add the object for the communication log, commlog object, to the work order resource, find the record for the oslcwodetail object structure. Add the commlog object as a source object. Specify WORKORDER as the parent object and specify COMMLOG as the relationship to the resource. In the OSLC Resources application, find the WORKORDER resource and verify that the COMMLOG object is added.
2. Import the updated OSLC resource from Maximo Asset Management into Maximo Anywhere.
   a. Edit the `anywhere-rdfs-puller.xml` file and add the RDF name.
   b. Run the `anywhere-rdfs-puller.xml` file to import the RDF into the OSLC resources directory in Maximo Anywhere.
   c. Verify that the RDF is imported in the `Anywhere\MaximoAnywhere\oslc-docs\resources\rdf\oslc` directory.

   For example, in MobileFirst Studio edit the `anywhere-rdfs-puller.xml` file to add the commlog RDF.

```
<target name="all" description="downloads all rdfs" />
  <...>
 <downloadOneRdf context="/oslc/shapes/oslcalndomain"  />
 <downloadOneRdf context="/oslc/shapes/oslcwodetail/commlog" />
</target>
```

Run the `anywhere-rdfs-puller.xml` file to import the commlog RDF. Verify that the shape document for the commlog is imported into the `Anywhere\MaximoAnywhere\oslc-docs\resources\rdf\oslcshapes\oslcwodetail\commlog` directory.

3. Define the attribute and the resource for the related record in the Data section of the `app.xml` file.
   a. In MobileFirst Studio, open the `app.xml` file and add the attribute for the related record to the OSLC resource. The attribute defines the related record that is available to use in the app.
   b. Add the resource for the related record. The resource describes the metadata for the fields that the related record uses.

   For example, add the attribute for the commlog to the work order resource to show the communication log in a child view of the Work Order Details view for the Work Execution app. The Work Order Details view references the workOrder resource.

```
<resource providedBy="/oslc/sp/WorkManagement"
   describedBy="http://jazz.net/ns/ism/work/smarter_physical_infrastructure#WorkOrder"
   name="workOrder" pageSize="200" class="application.business.WorkOrderObject">
 <attributes>
  ....
  <attribute name="commloglist" describedByProperty="spi_wm:commlog"
     describedByResource="commLogResource" />
  ....
  <localAttribute name="commloglistsize" dataType="string"
     persistent="false" />
 </attributes>
 ....
</resource>
```

   Add the resource for the communication log with the fields to be used in the Work Execution app.

```
<!-- Comm Log Resource -->
<resource name="commLogResource">
  <attributes>
   <attribute name="createdate" describedByProperty="spi_wm:createdate" />
   <attribute name="createby" describedByProperty="spi_wm:createby" />
   <attribute name="subject" describedByProperty="spi_wm:subject" />
   <attribute name="message" describedByProperty="spi_wm:message" />
   <attribute name="sendfrom" describedByProperty="spi_wm:sendfrom" />
   <attribute name="sendto" describedByProperty="spi_wm:sendto" />
  </attributes>
</resource>
```

4. In the UI section of the `app.xml` file, create the view to show the details of the related record.
   a. Add a <view> element and specify the ID and label for the view.
   b. Define the resource for the view.
   c. Add the fields for the view.

   For example, to create the detail view for the communication log, specify the ID attribute as `WorkExecution.CommLogDetailView` in a new <view> element. The details view is called Communication Log Entry, and the following fields are added to the view as read-only fields:

   - **Created By**
   - **Send To**
   - **Subject**
   - **Message**

The date and time that the communication log was created is also added.

```
<!-- Define Comm Log Detail View -->
  <view id="WorkExecution.CommLogDetailView" label="Communication Log Entry">
   <requiredResources>
    <requiredResource name="workOrder">
     <requiredAttribute name="commloglist" />
    </requiredResource>
   </requiredResources>
   <container resource="workOrder" attribute="commloglist">
    <group>
     <groupitem>
      <text resourceAttribute="createdate" editable="false" />
     </groupitem>
     <groupitem>
      <text label="Created By" resourceAttribute="createby"
       editable="false" />
     </groupitem>
     <groupitem>
      <text label="Send To" resourceAttribute="sendto"
       editable="false" />
     </groupitem>
     <groupitem>
      <text label="Subject" resourceAttribute="subject" editable="false" />
     </groupitem>
     <groupitem>
      <text label="Message" resourceAttribute="message" editable="false" />
     </groupitem>

    </group>
    <group>
     <groupitem>
      <lastupdatetext />
     </groupitem>
    </group>
   </container>
  </view>
```

5. In the UI section of the `app.xml` file, create the view to show a list of related records for the mobile app.
   a. Add a <view> element and specify the ID and label for the view.
   b. Define the resource and attribute for the view.
   c. Add the fields for the view.
   d. Add a link to the details view.
   e. Specify the layout template name for the list view.

   For example, to create the list view for the communication log, specify the ID attribute as `WorkExecution.CommLogView` in a new <view> element. The list view is called Communications Log. The following fields are added to the view as read-only fields:
   - **Created By**
   - **Send To**
   - **Subject**

   The date and time that the communication log was created is also added.

   To access the Communication Log Entry details view from the Communications Log list view, enter the ID for the `WorkExecution.CommLogDetailView` to the transitionTo attribute. The layout for the view is called `CommLogListItem`.

```
<!-- Define Comm Log List View --
 <view id="WorkExecution.CommLogView" label="Communications Log" >
  <requiredResources>
   <requiredResource name="workOrder">
    <requiredAttribute name="commloglist" />
   </requiredResource>
  </requiredResources>
  <list resource="workOrder" attribute="commloglist"
   transitionTo="WorkExecution.CommLogDetailView">
   <sortOptions>
    <sortOption label="Created Date" >
     <sortAttribute name="createdate" direction="asc" />
    </sortOption>
    <sortOption label="Created By">
     <sortAttribute name="createby" direction="asc" />
    </sortOption>
   </sortOptions>
  <listItemTemplate layout="CommLogListItem">
    <listtext resourceAttribute="createdate" layoutInsertAt="item1" />
    <listtext resourceAttribute="createby" layoutInsertAt="item2" />
    <listtext resourceAttribute="sendto" layoutInsertAt="item3" />
    <listtext resourceAttribute="subject" layoutInsertAt="item4"
     cssClass="bold textappearance-medium" />
    </listItemTemplate>
   </list>
  </view>
```

6. In the `Anywhere\MaximoAnywhere\apps\`*`app_name`*`\artifact\layouts\templates\small` directory, create the layout template for the list view.

   For example, to define the layout for the Communications Log view, create a layout file called `CommLogListItem.xml` in the `Anywhere\MaximoAnywhere\apps\WorkExecution\artifact\layouts\templates\small` directory and specify the structure for the layout.

```
<layout width="100">
 <row>
  <column columnid="item1" colspan="4" />
  <column columnid="item2" colspan="4" />
  <column columnid="item3" colspan="4" />
 </row>
 <row>
  <column columnid="item4" colspan="12"  />
 </row>
</layout>
```

7. In the UI section of the `app.xml` file, add a field on the main view in the mobile app to access the related record.

   For example, to access the Communications Log list view from the Work Order Details view, add the list view to the <groupitem> element within the view.

```
<view id="WorkExecution.WorkDetailView" label="Work Order Details" resource="workOrder">
 ....
 <container resource="workOrder">
 ....
  <group>
   <groupitem transitionTo="WorkExecution.CommLogView" layout="Item1Count1Button1">
     <text value="Communications Log" editable="false" layoutInsertAt="item1"
      cssClass="relatedRecords" />
     <text resourceAttribute="commloglistsize" editable="false"
      layoutInsertAt="count1">
     </text>
   </groupitem>
  </group>

  <group>
   <groupitem>
    <lastupdatetext />
   </groupitem>
  </group>
 </container>
</view>
```

8. Optional: To show the number of related records that are available from the main view, specify the variables for the list size and the attribute of the related record in the associated JavaScript file in the `MaximoAnywhere\apps\WorkExecution\common\js\application\handlers` directory.

a. Specify the resource attribute of the related record in the **attributes** variable. The resource attribute for the related record must be defined in the OSLC resource in the app.xml file.

b. Specify the resource attribute for the list size of the related record in the **listSizeArray** variable. The resource attribute for the list size must be defined as a local attribute in the OSLC resource in the app.xml file.

For example, to show the number of communications logs for each work order on the Work Order Details view, add the following variables to the WODetailHandler.js file in the MaximoAnywhere\apps\WorkExecution\common\js\application\handlers directory:

- Add the commloglistsize resource attribute to the **listSizeArray** variable.
- Add the commloglist resource attribute to the **attributes** variable.

```
var listSizeArray = ['tasklistsize', 'assignmentlistsize', 'materiallistsize',
          'toollistsize', 'actuallaborlistsize', 'actualmateriallistsize',
           'actualtoollistsize','workloglistsize', 'multiassetloclistsize',
          'attachmentssize', 'commloglistsize'];
var attributes = ["tasklist", "assignmentlist", "materiallist", "toollist", "actuallaborlist",
       "actualmateriallist", "actualtoollist", "workloglist", "multiassetloclist",
       "attachments", "commloglist"];
```

9. Save your changes. The app is built automatically.

What to do next

To verify your changes in a mobile browser simulator, right-click the application folder and select **Run As** > **Preview**.

Deploy the app to the server

Organizing the layout of mobile screens

You can arrange the layout of the mobile screens for the Maximo Anywhere apps to match the process and work flow of your organization. You can change the order of fields on a screen and modify layout template files to accommodate new fields.

Layout template files

A layout template file is an XML file that defines the structure and the sequence of controls on views in a Maximo Anywhere mobile app. When you change the structure of a view in the application definition file, you must update the associated layout template file.

Maximo Anywhere include layout template files for the list views of the mobile apps. In the app.xml files for the mobile apps, the <listitemTemplate> element identifies the name of the layout template file that is used for the view.

The layout template files for the Work Execution app are in the Anywhere\MaximoAnywhere\apps\WorkExecution\artifact\layouts\templates\small directory and for the Work Approval app is in the Anywhere\MaximoAnywhere\apps\WorkApproval\artifact\layouts\templates\small directory.

Structure of layout files

The <layout> element is the root element of a layout template file. The <row> element and the <column> element within the layout identify the placement of the fields in the view. To apply a structure to a mobile app, you can define the following attributes:

width attribute
: The width of the layout as a percentage on the mobile screen. You define the width in the <layout> element.

columnid attribute
: The attachment point for the field in the mobile view. The columnid attribute is associated with the layoutInsertAt attribute in the `app.xml` file.

colspan attribute
: The number of columns that the row occupies.

rowspan attribute
: The number of rows a column should span. You adjust the rowspan if you need to maintain the vertical position of columnid attributes.

Example of the layout for the work list view

The `WorkListItem.xml` layout file has three rows that define the structure of any associated mobile view and spans the width of the mobile screen. Each row defines the columnid attribute that is associated with the layoutInsertAt attribute in the `app.xml` file. The colspan attribute defines the number of columns that the row occupies.

```
<layout width="100">
 <row>
  <column columnid="item3" colspan="10" />
  <column columnid="button1" colspan="2" rowspan="3" halign="right"/>
 </row>
 <row>
  <column columnid="item1" colspan="3" />
  <column columnid="item2" colspan="7" />
 </row>
 <row>
  <column columnid="item4" colspan="5"  />
  <column columnid="item5" colspan="5" />
 </row>
</layout>
```

The Assigned Work view is defined in the `app.xml` file for the Work Execution app. The associated layout file, which is identified by the layout attribute, is the `WorkListItem.xml` file. The resourceAttribute defines each field that is shown in the view. The layoutInsertAt attribute defines the attachment point for the fields. For example, the layoutInsertAt attribute for the startime field is identified as `item 2`. The columnid attribute in the `WorkListItem.xml` file for `item2` is in the second row. Therefore, the start time field is on the second row of the view.

```
<view id="WorkExecution.WorkItemsView" label="Assigned Work"
   saveonshow="true" showBackButton="false" >
 <listItemTemplate layout="WorkListItem">
  <listtext resourceAttribute="wonum" layoutInsertAt="item1"
   cssClass="bold textappearance-medium"/>
   <listtext resourceAttribute="starttime" layoutInsertAt="item2"/>
  <listtext resourceAttribute="statusdesc" layoutInsertAt="item4"/>
  <listtext resourceAttribute="description" layoutInsertAt="item3"
   cssClass="bold textappearance-medium"/>
  <listtext resourceAttribute="assetnumanddescription" layoutInsertAt="item5"/>
</view>
```

Changing the layout of fields in list views

List views, which display records in a scrollable list, have a layout that you can customize. For example, you might want to change the order of existing fields or add a field. When you change the order of a view in the `app.xml` file, you apply the changes to the associated layout template file also.

About this task

In the `app.xml` file, the name of the layout template that is associated with a view is identified in the layout attribute of the <listItemTemplate> element. The <listtext> element defines the resource attribute or field within the structure of the layout and the layoutInsertAt attribute defines the attachment point of the field in the mobile view. In the layout template file, the value of the columnid attribute corresponds to the value of the layoutInsertAt attribute.

If you change the structure of a view, the value of the columnid attribute and the layoutInsertAt attribute must match for the changes of the layout in the view to take effect. For example, if you add a field to the view in the `app.xml` file, the value of the columnid attribute in the layout template file must match and the value of the layoutInsertAt attribute for the field to appear in the mobile view.

Procedure

1. In the UI section of the `app.xml` file, find the view that you want to change.
2. Modify the <listtext> element in the <listItemTemplate> element for the view by updating the resource attribute and the layoutInsertAt attribute and save the file.

   For example, to add a field named Supervisor to the work list view for Assigned Work, add the resource attribute for the field to the <listtext> element in the <listItemTemplate>. Identify the layoutInsertAt attribute as `item6`. You identify the layoutInsertAt attribute as `item6` because the supervisor field is the sixth field in the view.

   ```
   <view id="WorkExecution.WorkItemsView" label="Assigned Work"
      saveonshow="true" showBackButton="false" >
    <listItemTemplate layout="WorkListItem">
     <listtext resourceAttribute="wonum" layoutInsertAt="item1"
      cssClass="bold textappearance-medium"/>
      <listtext resourceAttribute="starttime" layoutInsertAt="item2"/>
     <listtext resourceAttribute="statusdesc" layoutInsertAt="item4"/>
     <listtext resourceAttribute="description" layoutInsertAt="item3"
      cssClass="bold textappearance-medium"/>
     <listtext resourceAttribute="assetnumanddescription" layoutInsertAt="item5"/>
     <listtext resourceAttribute="supervisor" layoutInsertAt="item6"/>
   </view>
   ```

   By adding the field, you must apply the change to the layout template file.

   Note: You must also add the resource data for the supervisor field to the Work Order resource for the field to exist in the view.
3. Open the associated layout template file in the `Anywhere\MaximoAnywhere\apps\`*`app_name`*`\artifact\layouts\templates\small` directory
4. Modify the layout structure to reflect the changes that you made in the `app.xml` file. You can make the following changes to update the layout:
   - Enter a <row> element to add a row.
   - Enter the columnid attribute to identify the attachment point of the field. The value for the columnid attribute must match the value of the layoutInsertAt attribute for the field.
   - Add a value for the colspan attribute to move the <column> element within the row to a specified position.

- Update the value of the rowspan attribute to adjust vertical positioning of any columnid attributes. If a new field was incorporated into an existing row, you do not need to adjust the rowspan attribute.

For example, the associated layout file for the work list view currently has four rows. The rowspan attribute for the `button1` columnid in the second row is `3` because the button spans across the three subsequent rows in the view.

```
<layout width="100">
  <row>
   <column columnid="item1" colspan="10" />
  </row>
  <row>
   <column columnid="item3" colspan="10" />
   <column columnid="button1" colspan="2" rowspan="3" halign="right"/>
  </row>
  <row>
   <column columnid="item2" colspan="10" />
  </row>
  <row>
   <column columnid="item4" colspan="5"  />
   <column columnid="item5" colspan="5" />
  </row>
</layout>
```

To include the addition of the supervisor field in the list view in a new row, add a <row> element to the layout and define the columnid attribute as `item6`. The rowspan for the `button1` columnid is `4` so that the timer button spans down the subsequent rows, including the new row, in the view.

```
<layout width="100">
  <row>
   <column columnid="item1" colspan="10" />
  </row>
  <row>
   <column columnid="item3" colspan="10" />
   <column columnid="button1" colspan="2" rowspan="4" halign="right"/>
  </row>
  <row>
   <column columnid="item2" colspan="10" />
  </row>
  <row>
   <column columnid="item4" colspan="5"  />
   <column columnid="item5" colspan="5" />
  </row>
  <row>
   <column columnid="item6" colspan="5" />
  </row>
</layout>
```

5. Save the changes to the layout. The app is built automatically.
6. Optional: To preview your changes, right-click on the application folder and select **Run As** > **Preview**.

What to do next

Deploy the app to the server.

Customizing javascript files for mobile apps

Maximo Anywhere applies business rules and behavior to the mobile apps. You can customize JavaScript files that are associated with the `app.xml` files to implement these rules.

Creating conditional fields in apps

To control when a field appears in a Maximo Anywhere app, you can make the field conditional.

About this task

You specify an event handler in the app.xml file to control whether to show or hide a field that depends on values of another field when the value is selected by the mobile user. Event handlers are implemented in JavaScript files that are associated with a view in the mobile app.

For example, if a work order has a work type of Emergency Maintenance, the **Risk** field should appear on the Work Order Details view. The **Risk** field is not needed for other types of work orders, so you need to apply a condition to control when the field appears. You create an event handler for the **Risk** field to monitor the resource and attribute that is associated with the work type and show the field if a mobile user selects Emergency Maintenance in the **Work Type** field. If a mobile user also changes the value for the work type, the handler checks the value to determine whether the **Risk** field is shown.

Procedure

1. In MobileFirst Studio, open the app.xml file for the app that you want to update and add the field that you want to apply the condition to.
2. In the UI section of the app.xml file, add a render type of event handler to the view that the fields belong to.
   a. In the <text> element of the new field, add a child element named <eventHandlers>.
   b. In the <eventHandlers> element, add a child element named <eventHandler>.
   c. Specify render as the event attribute, and add the method and class attributes.

   For example, to show the **Risk** field for emergency work types, add an event handler to the <groupitem> element for the **Risk** field in the Work Order Details view where the event attribute is render and the method attribute is handleConditionalRisk.

```
<view id="WorkExecution.WorkDetailView" label="Work Order Details">
...
 <groupitem>
  <text resourceAttribute="risk" label="Risk"
   editable="true" placeHolder="Tap to enter" />
    <eventHandlers>
       <eventHandler event="render" method="handleConditionalRisk"
       class="application.customerExtensions.WODetailExtensionHandler" />
    </eventHandlers>
  </text>
 </groupitem>
....
</view>
```

3. Apply the condition for the event handler to a JavaScript file.
   a. Create a folder for the handler in the MaximoAnywhere\apps\app_name\common\js\application directory. The folder name must match the name that is applied to the class attribute of the event handler.
   b. Add the JavaScript file to the MaximoAnywhere\apps\app_name\common\js\application directory. The name of the JavaScript file must match the name that is applied to the class attribute of the event handler.
   c. Add the condition for the render event handler to the JavaScript file.

   For example, to add the JavaScript code for the event handler to show the **Risk** field for emergency work types, create the WODetailExtensionHandler.js file in the MaximoAnywhere\apps\app_name\common\js\application\customerExtensions directory and add the JavaScript code.

```
define("application/customerExtensions/WODetailExtensionHandler",
     [ "dojo/_base/declare",
       "dojo/_base/lang",
       "platform/handlers/_ApplicationHandlerBase",
       "application/handlers/CommonHandler"],
function(declare, lang, ApplicationHandlerBase, CommonHandler) {

 return declare( [ApplicationHandlerBase],  {


  handleConditionalRisk: function(eventContext){
   // This is an render handler for the WO Details View
   // to handle changes made to the worktype field and show the Risk field

   // current workOrder is on the eventContext
   var currWO = CommonHandler._getAdditionalResource(eventContext,"workOrder").getCurrentRecord();
   var worktype = currWO.get('worktype');

   // The current risk field is on the eventContext
   // to show the risk field if worktype equals to EMERGENCY
   eventContext.setDisplay(worktype && worktype == "EM");

   // Hook a listener to watch the worktype attribute,
   // and make the risk field appear when worktype is Emergency
   if (!eventContext.hasResourceWatch("workTypeWatch")) {
       eventContext.addResourceWatchHandle(currWO.watch('worktype',
     lang.hitch(this, function(attrName, oldValue, newValue)
        {
         eventContext.setDisplay(newValue && newValue=="EM");
        }
    )),"workTypeWatch");
   }
  }

 });
});
```

4. Save your changes and preview the updated mobile app in a mobile browser simulator.

What to do next

Deploy the app to the server

Related tasks:

"Adding fields to mobile apps" on page 14
The fields that you add to mobile apps are defined by OSLC Resource Description Framework (RDF) files. When you add fields to mobile apps, you specify the OSLC resource for the field as a resource attribute in the application definition file.

Creating read-only conditional fields

To control when a field becomes read-only or read/write dynamically in a Maximo Anywhere app, you can make the field conditional to be read-only.

About this task

You specify an event handler in the `app.xml` file to control whether a field that depends on values of another field is read-only. Event handlers are implemented in JavaScript files that are associated with a view in the mobile app.

For example, if a work order has a work type of Emergency Maintenance, the **Priority** field should become read-only. The **Priority** field is not needed for other types of work orders, so you need to apply a condition to control when the field is read-only. You create an event handler for the **Priority** field to monitor the resource and attribute that is associated with the work type and make the field read-only if a mobile user selects Emergency Maintenance in the **Work Type** field. If a mobile user changes the value for the work type, the handler also checks the value to determine whether the **Priority** field is read-only.

Procedure

1. In MobileFirst Studio, open the `app.xml` file for the app that you want to update and add the field that you want to apply the condition to.
2. In the UI section of the `app.xml` file, add an initializer type of event handler to the fields.
   a. In the <text> element of the new field, add a child element named <eventHandlers>.
   b. In the <eventHandlers> element, add a child element named <eventHandler>.
   c. Specify `initialize` as the event attribute, and add the method and class attributes.

   For example, to make the **Priority** field read-only for emergency work types, add an initializer type of event handler to the view that the fields belong to. The event that is associated with the handler for the Work Order Details view is named WODetailExtensionHandler.

```
<view id="WorkExecution.WorkDetailView" label="Work Order Details">
...
  <eventHandlers>
   <eventHandler event="initialize" method="fetchAllListSizes"
    class="application.handlers.WODetailHandler" />

   <eventHandler event="initialize" method="handleConditionalReadOnlyPriority"
    class="application.customerExtensions.WODetailExtensionHandler" />

   <eventHandler event="render" class="application.handlers.WODetailHandler"
    method="refreshAllListSizes" />
   <eventHandler event="initialize"
    class="application.handlers.MetersListHandler" method="initializeMeters" />
  </eventHandlers>
....
</view>
```

3. Apply the condition for the event handler to a JavaScript file.
   a. Create a folder for the handler in the `MaximoAnywhere\apps\`*`app_name`*`\common\js\application` directory. The folder name must match the name that is applied to the class attribute of the event handler.
   b. Add the JavaScript file to the `MaximoAnywhere\apps\`*`app_name`*`\common\js\application` directory. The name of the JavaScript file must match the name that is applied to the class attribute of the event handler.
   c. Add the condition for the initialize event handler to the JavaScript file.

   For example, to add the condition for the event handler to make the **Priority** field read-only for emergency work types, create the `WODetailExtensionHandler.js` file in the `MaximoAnywhere\apps\`*`app_name`*`\common\js\application\customerExtensions` directory and add the JavaScript code.

```
define("application/customerExtensions/WODetailExtensionHandler",
      [ "dojo/_base/declare",
        "dojo/_base/lang",
        "platform/handlers/_ApplicationHandlerBase",
        "application/handlers/CommonHandler"],
function(declare, lang, ApplicationHandlerBase, CommonHandler) {

 return declare( [ApplicationHandlerBase],  {

  handleConditionalReadOnlyPriority: function(eventContext){
   // This is an initialize handler for the WO Details View
   // to handle changes made to the worktype field
   // and make the Priority field read-only

   // current workOrder is on the eventContext
   var currWO = eventContext.getResource().getCurrentRecord();
   var worktype = currWO.get('worktype');
```

```
    // Makes the priority field read-only if worktype equals EMERGENCY
    currWO.getRuntimeFieldMetadata('priority').set('readonly', worktype && worktype=="EM");

    // Hook a listener to watch the worktype attribute,
    // and make the priority attribute read-only when the worktype is Emergency
     eventContext.addResourceWatchHandle(currWO.watch('worktype',
     lang.hitch(this, function(attrName, oldValue, newValue)
        {
         currWO.getRuntimeFieldMetadata('priority').set('readonly', newValue && newValue=="EM");
        }
    )));
   }

 });
});
```

4. Save your changes and preview the updated mobile app in a mobile browser simulator.

What to do next

Deploy the app to the server

Creating required conditional fields

You can make a field required dynamically in a Maximo Anywhere app, depending on the values of other fields.

About this task

You specify an event handler in the `app.xml` file to control whether a field that depends on values of another field is required. Event handlers are implemented in JavaScript files that are associated with a view in the mobile app.

For example, if a work order has a work type of Emergency Maintenance, the **Priority** field should become required. The **Priority** field is not needed for other types of work orders, so you need to apply a condition to control when the field is required. You create an event handler for the **Priority** field to monitor the resource and attribute that is associated with the work type and make the field required if a mobile user selects Emergency Maintenance in the **Work Type** field. If a mobile user changes the value for the work type, the handler also checks the value to determine whether the **Priority** field is required.

Procedure

1. In MobileFirst Studio, open the `app.xml` file for the app that you want to update and add the field that you want to apply the condition to.
2. In the UI section of the `app.xml` file, add an initializer type of event handler to the fields.
   a. In the <text> element of the new field, add a child element named <eventHandlers>.
   b. In the <eventHandlers> element, add a child element named <eventHandler>.
   c. Specify `initialize` as the event attribute, and add the method and class attributes.

   For example, to make the **Priority** field required for emergency work types, add an initializer type of event handler to the view that the fields belong to. The event that is associated with the handler for the Work Order Details view is named WODetailExtensionHandler.

```
<view id="WorkExecution.WorkDetailView" label="Work Order Details">
...
  <eventHandlers>
   <eventHandler event="initialize" method="fetchAllListSizes"
    class="application.handlers.WODetailHandler" />

   <eventHandler event="initialize" method="handleConditionalRequiredPriority"
    class="application.customerExtensions.WODetailExtensionHandler" />

   <eventHandler event="render" class="application.handlers.WODetailHandler"
    method="refreshAllListSizes" />
   <eventHandler event="initialize"
    class="application.handlers.MetersListHandler" method="initializeMeters" />
  </eventHandlers>
....
</view>
```

3. Apply the condition for the event handler to a JavaScript file.
   a. Create a folder for the handler in the `MaximoAnywhere\apps\`*`app_name`*`\common\js\application` directory. The folder name must match the name that is applied to the class attribute of the event handler.
   b. Add the JavaScript file to the `MaximoAnywhere\apps\`*`app_name`*`\common\js\application` directory. The name of the JavaScript file must match the name that is applied to the class attribute of the event handler.
   c. Add the condition for the initialize event handler to the JavaScript file.

   For example, to add the condition for the event handler to make the **Priority** field required for emergency work types, create the `WODetailExtensionHandler.js` file in the `MaximoAnywhere\apps\`*`app_name`*`\common\js\application\customerExtensions` directory and add the JavaScript code.

```
define("application/customerExtensions/WODetailExtensionHandler",
      [ "dojo/_base/declare",
        "dojo/_base/lang",
        "platform/handlers/_ApplicationHandlerBase",
        "application/handlers/CommonHandler"],
function(declare, lang, ApplicationHandlerBase, CommonHandler) {

 return declare( [ApplicationHandlerBase],  {

  handleConditionalRequiredPriority: function(eventContext){
   // This is an initialize handler for the WO Details View
   // to handle changes made to the worktype field
   // and make the Priority field required

   // current workOrder is on the eventContext
   var currWO = eventContext.getResource().getCurrentRecord();
   var worktype = currWO.get('worktype');

   // Makes the priority field required if worktype equals EMERGENCY
   currWO.getRuntimeFieldMetadata('priority').set('required', worktype && worktype=="EM");

   // Hook a listener to watch the worktype attribute,
   // and make the priority attribute required when the worktype is Emergency
    eventContext.addResourceWatchHandle(currWO.watch('worktype',
    lang.hitch(this, function(attrName, oldValue, newValue)
      {
       currWO.getRuntimeFieldMetadata('priority').set('required', newValue && newValue=="EM");
      }
   )));
  }

 });
});
```

4. Save your changes and preview the updated mobile app in a mobile browser simulator.

What to do next

Deploy the app to the server

Notices

IBM Online Privacy Statement

IBM Software products, including software as service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s name, user name, password, or other personally identifiable information for purposes of session management, authentication, single sign-on configuration or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also likely eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details in the section entitled “Cookies, Web Beacons and Other Technologies” and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Index

Printed in USA