

Sterling Selling and Fulfillment Foundation



Customizing User Interfaces for Mobile Devices

Version 91

Sterling Selling and Fulfillment Foundation



Customizing User Interfaces for Mobile Devices

Version 91

Note

Before using this information and the product it supports, read the information in "Notices" on page 47.

Copyright

This edition applies to the 9.1 Version of IBM Sterling Selling and Fulfillment Foundation and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 1999, 2011.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Checklist for Customization

Projects	1
Customization Projects	1
Prepare Your Development Environment	1
Plan Your Customizations	1
Extend the Database	1
Make Other Changes to APIs	2
Customize the UI	2
Extend Transactions	3
Build and Deploy your Customizations or Extensions	3

Chapter 2. Planning Custom Interfaces for Mobile Devices 5

Differences between Console User and Mobile Device Interfaces	5
Guidelines for Smooth Upgrades and Maintenance	5
Design Guidelines for Mobile Device Screens	6
Planning Mobile Device Screen Size	6
Passing Data Between Mobile Device Screens	8
Error Handling for Mobile Devices	8

Chapter 3. Creating Resources for Mobile Device Interfaces 11

Creating Resources in the Applications Manager	11
Inventory Inquiry Resources in the Applications Manager: An Example	11

Chapter 4. Menu Entries and HTML Templates for Mobile Devices 17

Adding a Menu Entry	17
Creating an HTML Template	19

Chapter 5. Creating JSP Files 21

JSP File Structure	21
JSP File Name and Directory Guidelines	21

Chapter 6. Style Reference for Mobile Device Interfaces 23

HTML Tags for Mobile Device Interfaces	23
JSP Tag Library	24
Data Type Reference	24

Chapter 7. Programming Standards for Mobile Device Interfaces 25

JSP File Standards for Mobile Device Interfaces	25
Internationalization Standards for Mobile Device Interfaces	25
Validating HTML Files for Mobile Device Interfaces	25

Chapter 8. JSP Functions for Mobile Device Interfaces 27

JSP Functions Used for Mobile Device Interfaces	27
addToTempQ	27
clearTempQ	28
deleteAllFromTempQ	28
deleteFromTempQ	29
getErrorXML	29
getField	30
getForm	30
getStoredElement	31
getTempQ	31
getTempQValue	32
replaceInTempQ	32
resetAttribute	33
sendForm	33

Chapter 9. Configuring the Mobile Application 35

Configuring the Mobile Application	35
Configuring the Mobile Application Screens	36
New Attributes	36
Associating Resource ID with a Process	37
Use Cases	38

Chapter 10. Menu-Level Customization of the Mobile Application 43

Menu-Level Customization of the Mobile Application	43
Menu-Level Customization	43
Avoiding Copying of HTML Files	43
Avoiding Copying of JSP Files	43

Chapter 11. Configuring Mobile Application User Interface Components 45

Notices 47

Chapter 1. Checklist for Customization Projects

Customization Projects

Projects to customize or extend Sterling Selling and Fulfillment Foundation vary with the type of changes that are needed. However, most projects involve an interconnected series of changes that are best carried out in a particular order. The checklist identifies the most common order of customization tasks and indicates which guide in the documentation set provides details about each stage.

The items identified for extension and/or modification in the documentation are Source Components (to the extent such item involves source code) and Sample Materials for purposes of the License Information file associated with this product.

Prepare Your Development Environment

Set up a development environment that mirrors your production environment, including whether you deploy your application on a WebLogic, WebSphere®, or JBoss application server. Doing so ensures that you can test your extensions in a real-time environment.

You install and deploy your application in your development environment following the same steps that you used to install and deploy it in your production environment. Refer to your system requirements and installation documentation for details.

You have an option to customize your application with Microsoft COM+. Using Microsoft COM+ has advantages such as increased security, better performance, increased manageability of server applications, and support for clients of mixed environments. If this is your choice, see the *Customization Basics Guide* about additional installation instructions.

Plan Your Customizations

Are you adding a new menu entry? Or customizing the sign-in screen or logo? Or customizing views or wizards? Or creating new themes or new screens? Each type of customization varies in scope and complexity.

For background, see the *Customization Basics Guide*, which summarizes the types of changes that you can make and provides important guidelines about file names, keywords, and other general conventions.

Extend the Database

For many customization projects, the first task is to extend the database so that it supports the other UI or API changes that you make later. For instructions, see the *Extending the Database Guide*, which includes information about the following topics:

- Important guidelines about what you can and cannot change in the database.

- Information about modifying APIs. If you modify database tables so that any APIs are impacted, you must extend the templates of those APIs or you cannot store or retrieve data from the database. This step is required if table modifications impact an API.
- How to generate audit references so that you improve record management by tracking records at the entity level. This step is optional.

Make Other Changes to APIs

Your application can call or invoke standard APIs or custom APIs. For background about APIs and the services architecture of service types, behavior, and security, see the *Customizing APIs Guide*. This guide includes information about the following types of changes:

- Invoke standard APIs for displaying data in the UI and for saving changes made in the UI to the database.
- Invoke customized APIs for executing your custom logic in the extended service definitions and pipeline configurations.
- APIs use input and output XML to store and retrieve data from the database. If you don't extend these API input and output XML files, you may not get the results you want in the UI when your business logic is executing.
- Every API input and output XML file has a DTD and XSD associated to it. Whenever you modify input and output XML, you must generate the corresponding DTD and XSD to ensure data integrity. If you don't generate the DTD and XSD for extended XMLs, you may get inconsistent data.

Customize the UI

IBM® applications support several UI frameworks. Depending on your application and the customizations you want to make, you may work in only one or in several of these frameworks. Each framework has its own process for customizing components such as menu items, logos, themes, and so on.

Depending on the framework you want, consult one of the following guides:

- *Customizing the Console JSP Interface Guide*
- *Customizing the Swing Interface Guide*
- *Customizing User Interfaces for Mobile Devices Guide*
- *Customizing the Rich Client Platform Guide* and *Using the RCP Extensibility Tool Guide*
- *Customizing the Web UI Framework Guide*

Depending on the framework you want, consult one of the following guides:

- *Customizing the Console JSP Interface Guide*
- *Customizing the Swing Interface Guide*
- *Customizing User Interfaces for Mobile Devices Guide*
- *Customizing the Rich Client Platform Guide* and *Using the RCP Extensibility Tool Guide*
- *Customizing the Web UI Framework Guide*

Extend Transactions

You can extend and enhance the standard functionality of your application by extending the Condition Builder and by integrating with external systems. For background about transaction types, security, dynamic variables, and extending the Condition Builder, see the *Extending Transactions Guide* and *Extending the Condition Builder Guide*. These guides includes information about the following types of changes:

- Extend the Condition Builder to define complex and dynamic conditions for executing your custom business logic and using a static set of attributes.
- Define variables to dynamically configure properties belonging to actions, agents, and services configurations.
- Set up transactional data security for controlling who has access to what data, how much they can see, and what they can do with it.
- Create custom time-triggered transactions. You can invoke and schedule custom time-triggered transactions in much the same manner as you invoke and schedule the time-triggered transactions supplied by your application.
- Coordinate your custom, time-triggered transactions with external transactions and run them either by raising an event, calling a user exit, or invoking a custom API or service.

Build and Deploy your Customizations or Extensions

After performing the customizations that you want, you must build and deploy your customizations or extensions.

1. Build and deploy your customizations or extensions in the test environment so you can verify them.
2. When you are ready, repeat the same process to build and deploy your customizations and extensions in your production environment.

For instructions about this process, see the *Customization Basics Guide* which includes information about the following topics:

- Building and deploying standard resources, database extensions, and other extensions (such as templates, user exits, and Java interfaces).
- Building and deploying enterprise-level extensions.

Chapter 2. Planning Custom Interfaces for Mobile Devices

Differences between Console User and Mobile Device Interfaces

Sterling Selling and Fulfillment Foundation enables you to develop and display a custom user interface for the mobile devices used in warehouse operations.

Mobile device user interface extensibility is accomplished through scripts that determine how the user interface renders the screen and passes data.

Before beginning, you need to understand how to develop HTML, JSP, and XML components, how to use APIs, and how to use the Application Console and the Applications Manager user interfaces.

When customizing the interface, copy the standard resources of Sterling Selling and Fulfillment Foundation and then modify your copy, or create a completely new view. Do not modify the standard resources.

Note that the mobile device user interface differs from the Console user interface in the following ways:

- Mobile device screens use separate architecture for search and list views. If you need search view and list views functionality, model them as detail views.
- Mobile device screens can have only one detail view. Each detail view can contain only one inner panel.
- A mobile device inner panel cannot have any actions or icons.

The APIs return the data that needs to be displayed. For information on functions specific to mobile devices that are used within the JSP files, see “JSP Functions Used for Mobile Device Interfaces” on page 27

Customizing the mobile device user interface is accomplished using the Applications Manager user interface. For more information, see the Sterling Selling and Fulfillment Foundation: *Application Platform Configuration Guide*.

Guidelines for Smooth Upgrades and Maintenance

- Do not change the resource definitions of any of the resources shipped as part of the standard default configuration. Either make a copy through the Applications Manager and then change the copy, or create your own new views.
- Do not change any of the JSP files, JavaScript files and icon JAR files that are supplied by the application. If you do, your changes may be lost during upgrades.
- When creating new views, consider issues regarding ease of maintenance as well as ease of creation. When you create a new view, inner panel, and so forth, it is possible to link to the JSPs supplied by the application. But in future releases, the application may add more resources to these JSP, which means you must monitor software changes and update your configuration to account for these changes.
- Build in usability: Any new views you develop should look and behave like the product views, so before you begin developing, gain an understanding of how the default views behave.

- Prepare your development environment: In order to start the customization process, prepare the development environment to accommodate development and testing of the mobile user interface changes.

Design Guidelines for Mobile Device Screens

In order to optimize the display of data and execution of transactions, design simple screens and simple transactions using the following rules:

- Avoid placing a lot of information into a small space. This ensures more rapid transaction time and enables the end user to parse data visually more quickly.
- Because of the reduced screen size, if you need to display a lot of data, the display of data may need to be altered to accommodate the amount of data. In this case, the data must be persisted from one screen to the next before finally being posted. For small screens, use the TEMPQ utilities to pass data between screens.
- The TempQ utilities enables you to persist and pass data from one screen to the next. For information on the TempQ utilities, see “JSP Functions Used for Mobile Device Interfaces” on page 27
- Provide text on one line and the data field on another line to accommodate for internationalization requirements.
- Validate fields only when necessary in order to optimize transaction execution time.
- Choose fast or templated APIs that return exactly the correct type of information needed in order to optimize transaction execution time.

Planning Mobile Device Screen Size

Keep the following things in mind when defining the screen size dimensions for a mobile device:

- Restrict the screen size to 8 lines X 24 (or 22) characters per line. This ensures that your custom screen also displays correctly on a VT220 terminal.
- A mobile device screen can contain only table.
- A mobile device screen can handle a combined total of the following hidden and displayed fields:

Field	Maximum Size
Text and hidden fields	15
Labels and protected fields	15
Command buttons	5

- When the maximum allowed size for a given field is violated by a user (for example, when a user enters 20 hidden fields), the following error message is displayed: "An error was encountered while running this program: Invalid procedure call or argument."
- Draw a layout of each screen. For example, creating inventory screens requires an Inventory Inquiry screen and an Inventory Detail screen.

1	12345678901234567890
1	Inventory Inquiry
2	Enterprise []
3	Item ID []
4	Desc
5	Product Class []
6	UOM []
7	
8	
	[Back] [Next View] [Inquire]

Figure 1. Inventory Inquiry Screen

2	12345678901234567890
1	Enterprise []
2	Reference []
3	Item ID []
4	Desc
5	Product Class []
6	UOM []
7	Quantity 9999999999
8	
	[Back]

Figure 2. Inventory Detail Screen

Note the use of fixed width font while drawing the screen layouts. The buttons are not counted in the 8-row limit.

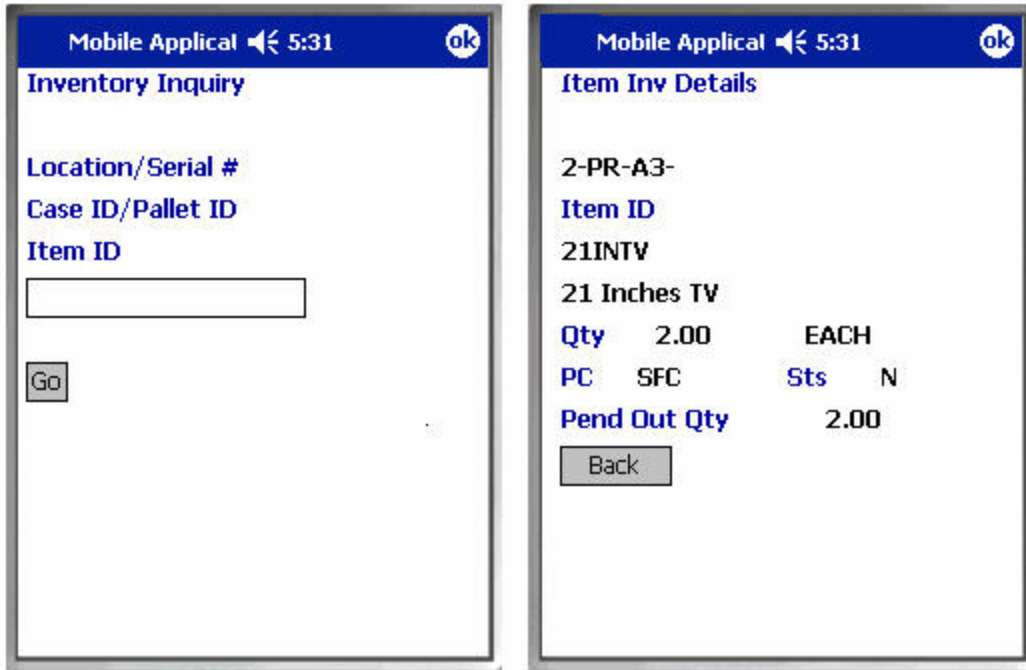


Figure 3. Inventory Inquiry Screen, Inventory Detail Screen

In this example, the `getItemList()` API fetches item details based on the information submitted on the Inventory Inquiry Search Screen, and `getATP()` API fetches inventory details for the item on the Inventory Inquiry Detail screen.

- List the APIs that must be called when each screen is navigated to. This should include the entire API input that is passed and the API template that is used for filtering the API output, if applicable.

Passing Data Between Mobile Device Screens

Because of the small screen size, the data may need to be persisted from one screen to the next before finally being posted. When passing data between screens, you can either use hidden text or the TempQ utilities. We recommend the usage of TempQ. A TempQ stores the name/value pair information on one page in the session and provides for accessor methods on the subsequent pages. For details on these utilities, see “JSP Functions Used for Mobile Device Interfaces” on page 27

Error Handling for Mobile Devices

Error handling is taken care of by the utility method `getError XML` as shown in the `rfutil.jspf` file. For information on the `getError XML`, see “`deleteFromTempQ`” on page 29

Validations performed by tabbing out of a field or by clicking a button may lead to an error. These errors are displayed in a standard error XML in the response output stream:

```
<errors>
  <error errortxt="errorDesc" focus="errorField"/>
</errors>
```

See the following Inventory Inquiry error message example.

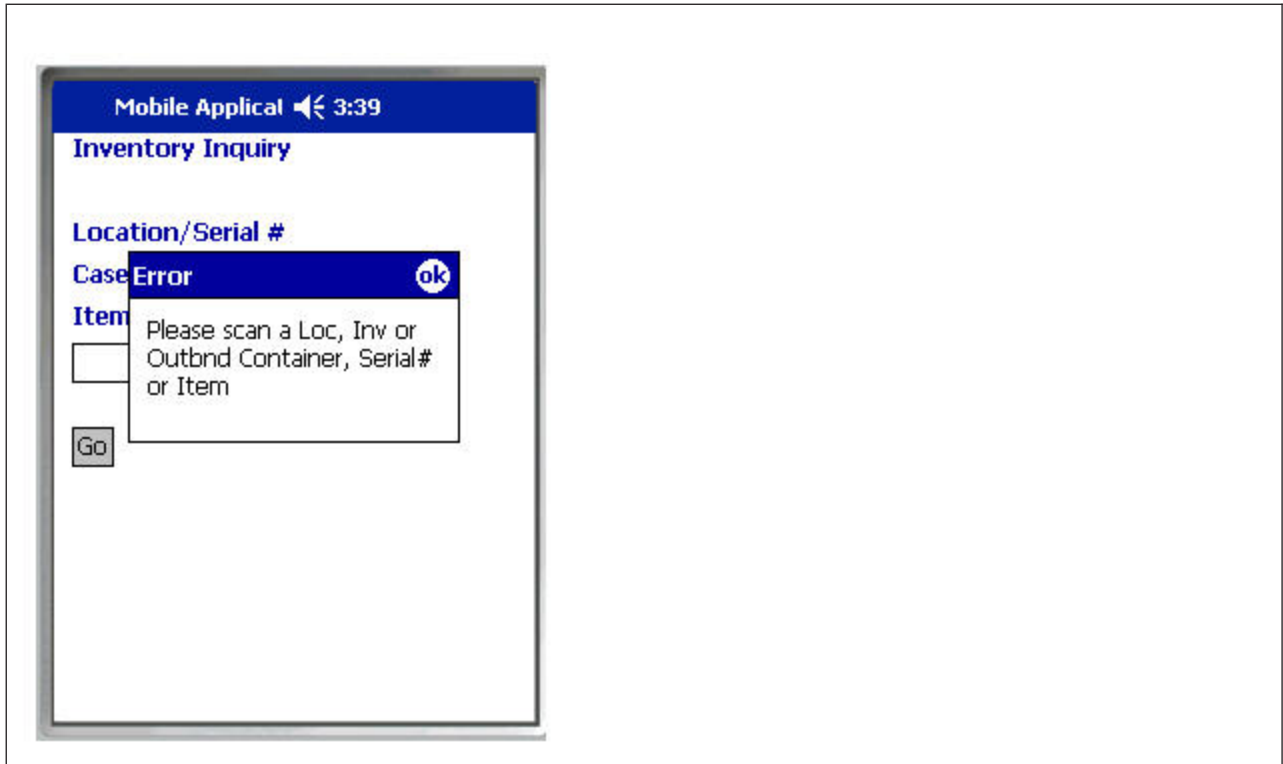


Figure 4. Error Condition 1: Item ID is not entered

Chapter 3. Creating Resources for Mobile Device Interfaces

Creating Resources in the Applications Manager

About this task

Mobile device screens are composed of screen resources, such as an entity, a detail view, inner panels, and APIs. The following mobile device resources define the screen look and feel, screen behavior, and screen flow:

Resource	Description
Screen Entity	Controls access to transactions. It also provides the starting point (JSP name).
Detail View	Mobile device screens are modeled as detail views. A screen can have only one detail view. Each detail view can contain only one inner panel. A screen can contain only tables.
Inner Panel	Mobile device inner panels cannot have any actions or icons.
APIs	Required APIs can be defined under the API List for the inner panel.

To create custom mobile device resources:

Procedure

Configure the resources described in the table.

Inventory Inquiry Resources in the Applications Manager: An Example

The Creating Inventory Inquiry screens requires the following user interface resources:

- A screen entity called rfinventory
- Detail view called rfinventoryD1
- Inner panel called rfinventoryIP1
- getItemDetails() API called rfinventoryD1IP1API

These resources are detailed in the following tables.

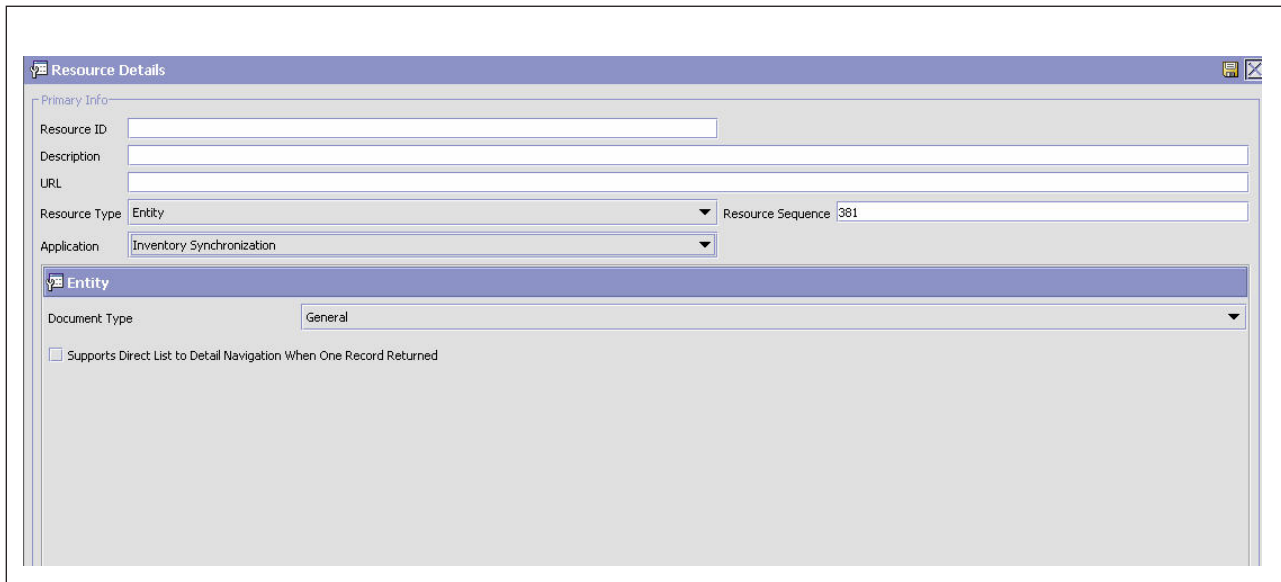


Figure 5. Mobile Device Screen Entity Resource

Table 1. Mobile Device Screen Entity Resource Values

Name	Value
Resource ID	rfinventory
Description	RF_Inventory
Resource Type	Entity
Resource Sequence	(Default Suggested Value)
Application	Warehouse Management
Document Type	General

The screenshot shows a 'Resource Details' window with the following fields and values:

- Resource ID: rfinventoryD1
- Description: RF_Inventory_Detail_View
- URL: (empty)
- Resource Type: Detail View
- Resource Sequence: 1
- Application: Warehouse Management
- Java Server Page: (empty)
- Output Name Space: (empty)
- Height: (empty)
- Width: (empty)
- Ignore Default API:
- Hide Title Bar:
- Hide Navigation Panel:
- This View Redirects To An Alternate View:
- View Group ID: (empty)
- Input: (empty)
- Template: (empty)

Figure 6. Mobile Device Screen Detail View

Table 2. Mobile Device Screen Detail View Resource Values

Name	Value
Resource ID	rfinventoryD1
Description	RF_Inventory_Detail_View
Resource Type	Detail View
Resource Sequence	(Default Suggested Value)
Application	Warehouse Management

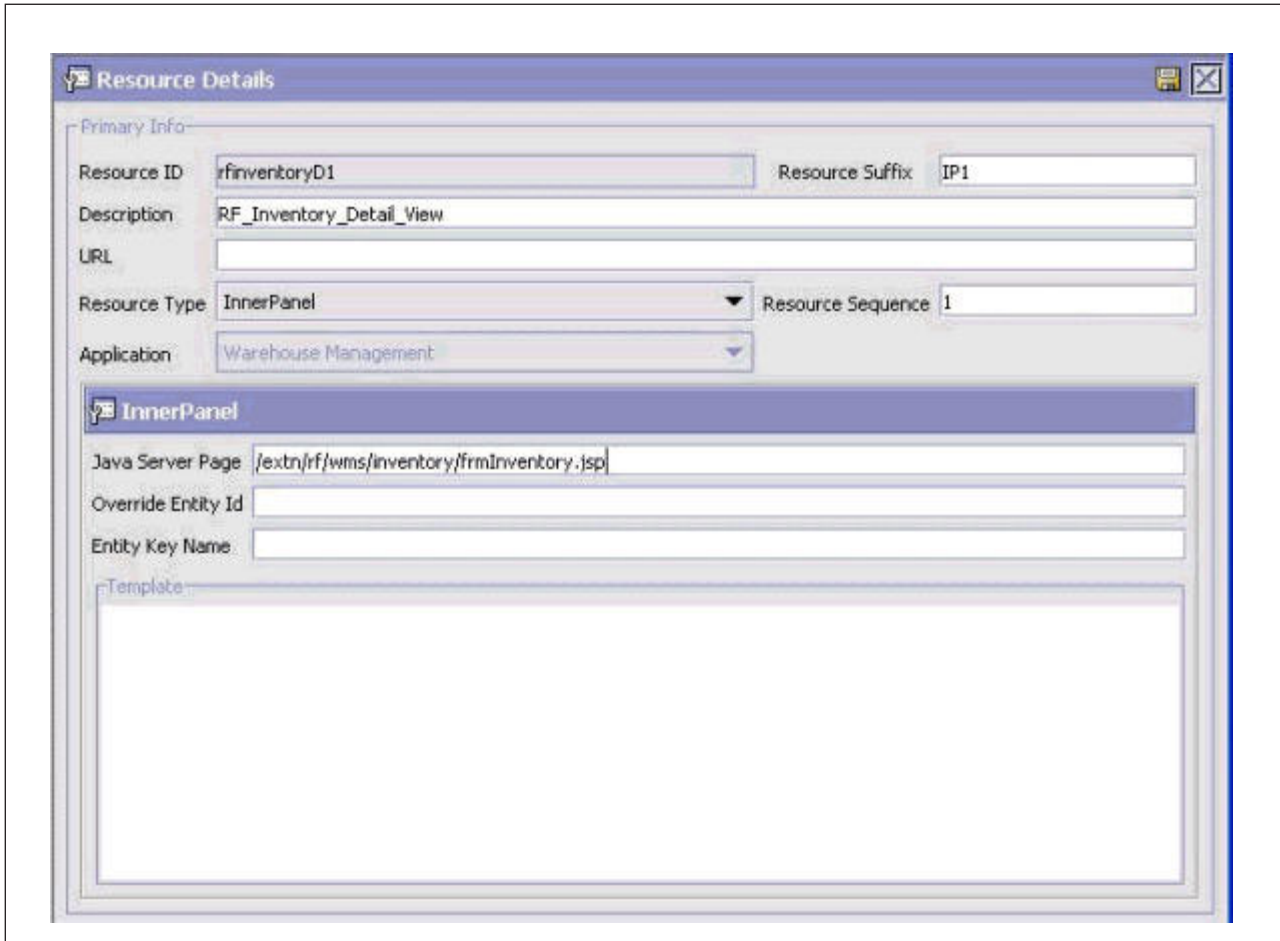


Figure 7. Mobile Device Screen Inner Panel Resource

Table 3. Mobile Device Screen Inner Panel Resource Values

Name	Value
Resource ID	rfinventoryD1IP1
Description	RF_Inventory_Inner_Panel
Resource Type	Inner Panel
Resource Sequence	(Default Suggested Value)
Application	Warehouse Management
Java Server Page	/extensions/global/webpage/rf/wms/inventory/frmInventory.jsp

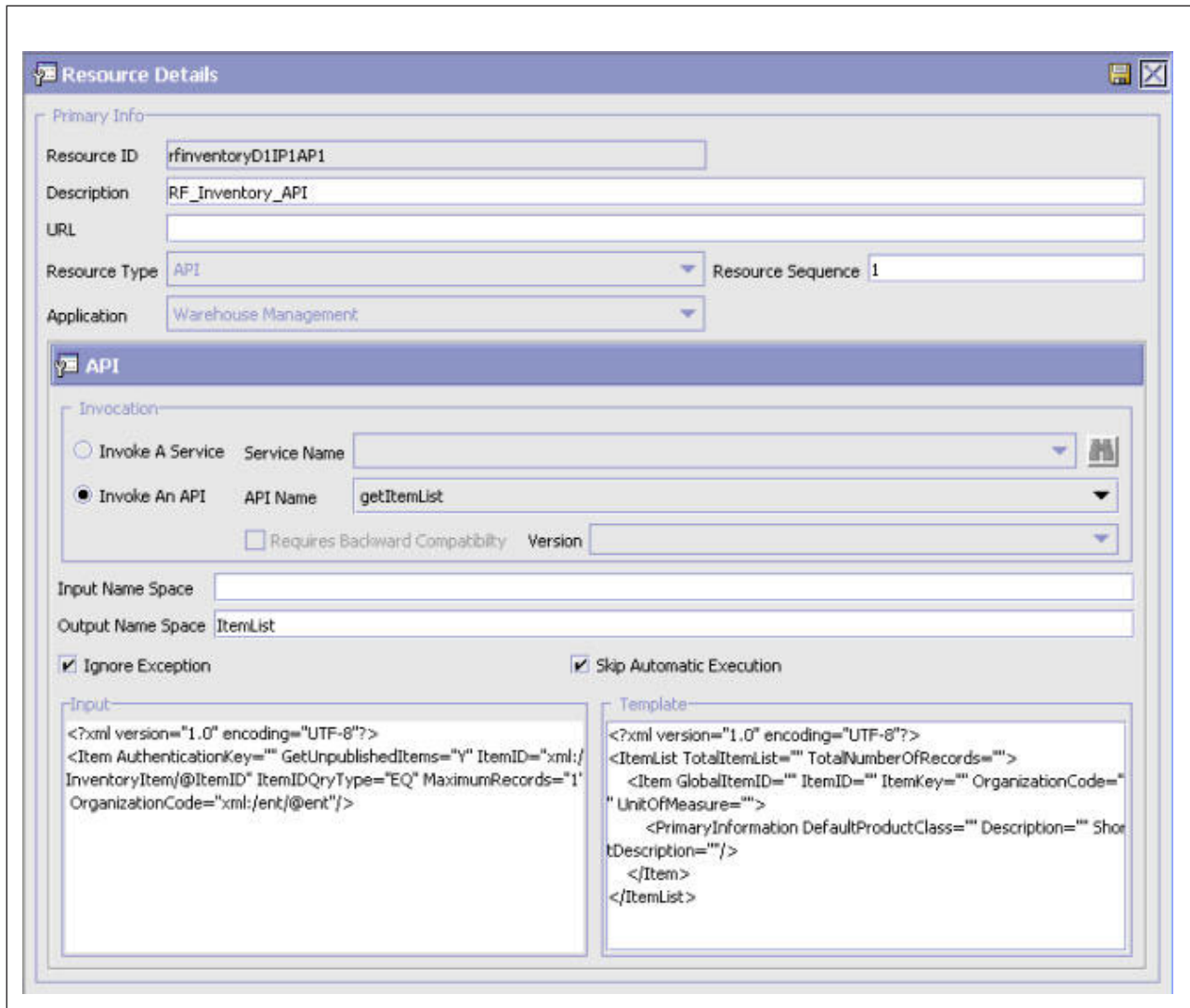


Figure 8. Mobile Device Screen API Resource

Table 4. Mobile Device Screen API Resource Values

Name	Value
Resource ID	rfinventoryD1IP1AP1
Description	RF_Inventory_API
Resource Type	API
Resource Sequence	(Default Suggested Value)
Application	Enter Warehouse Management.
Invoke an API	Select this radio button.
API Name	getItemList
Output Name Space	ItemList
Ignore Exception	Select this checkbox
Skip Automatic Execution	Always select this checkbox when defining a mobile device screen API resource.

Table 4. Mobile Device Screen API Resource Values (continued)

Name	Value
Input	<pre data-bbox="643 258 1419 359"><?xml version="1.0" encoding="UTF-8"?> <Item AuthenticationKey="" GetUnpublishedItems="Y" ItemID="xml:/InventoryItem/@ItemID" ItemIDQtyType="EQ" MaximumRecords="1" OrganizationCode="xml:/ent/@ent"/></pre>
Template	<pre data-bbox="643 380 1419 588"><?xml version="1.0" encoding="UTF-8"?> <ItemList TotalItemList="" TotalNumberOfRecords=""> <Item GlobalItemID="" ItemID="" ItemKey="" OrganizationCode="" UnitOfMeasure=""> <PrimaryInformation DefaultProductClass="" Description="" ShortDescription=""/> </Item> </ItemList></pre>

Chapter 4. Menu Entries and HTML Templates for Mobile Devices

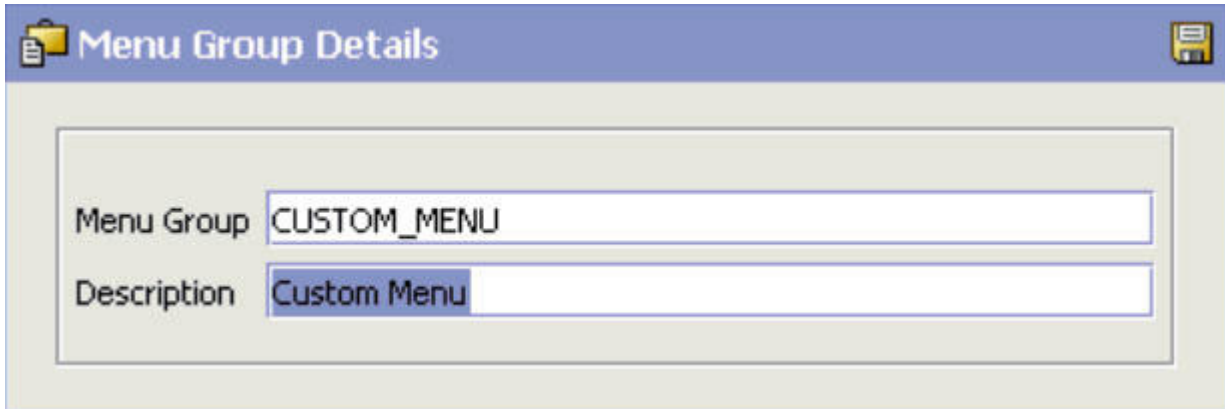
Adding a Menu Entry

About this task

To create a mobile device menu entry:

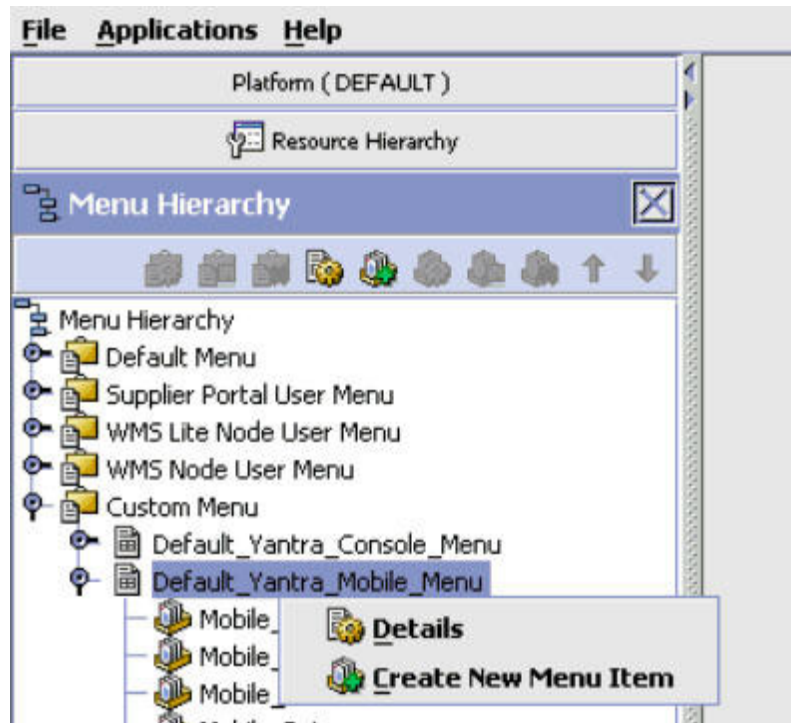
Procedure

1. Create a menu entry using the Menu Applications Manager. On saving the screen, a new Menu Group called “Custom Menu” is created.



The screenshot shows a dialog box titled "Menu Group Details" with a blue header bar. Inside the dialog, there are two text input fields. The first field is labeled "Menu Group" and contains the text "CUSTOM_MENU". The second field is labeled "Description" and contains the text "Custom Menu". The dialog box has a standard Windows-style border and a save icon in the top right corner.

2. Right-click **Custom Menu** → **Default_Yantra_Mobile_Menu** → **Create New Menu Item** and select **Details**.



3. Add a new Menu Item with the description Mobile_Inventory_Inquiry, the Resource ID rfinventory, and the menu sequence as suggested on this Menu Item Details screen:

Menu Item Details
[Save]

Menu Id	<input type="text" value="200401221713229640"/>
Description	<input type="text" value="Mobile_Inventory_Inquiry"/>
Icon	<input type="text"/>
Menu Sequence	<input type="text" value="12"/>
Resource ID	<input type="text" value="rfinventory"/> [Image Icon]

4. Create a bundle property for Mobile_Inventory_Inquiry with a value 'Inventory Inquiry' in the *INSTALL_DIR/extensions/global/resources/extnbundle.properties* file. This creates a mobile device menu option called Inventory Inquiry.

Note: Ensure that the following file does not exist: *INSTALL_DIR/resources/extn/extnbundle.properties*

This file must be removed because it will conflict with the extensions build process for bundle entries.

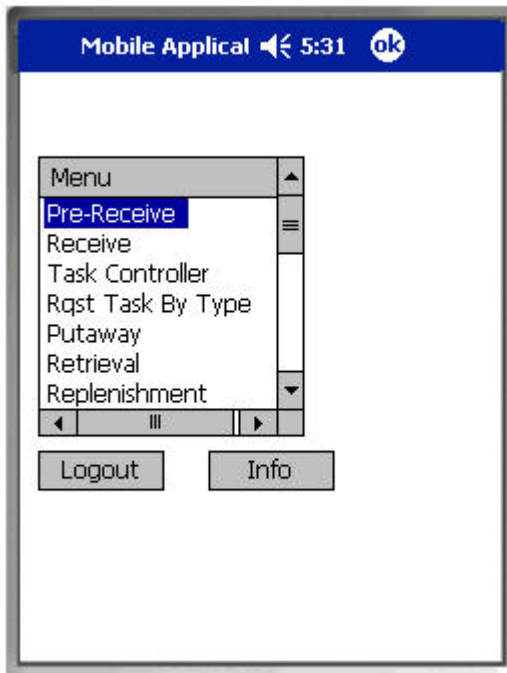


Figure 9. Mobile Application Menu Screen

Choosing this Menu option eventually invokes the JSP defined in the inner panel associated with this entity.

For example, in the `rfinventory` UI entity, the `/extensions/global/webpages/rf/wms/inventory/fmInventory.jsp` file relative to the Sterling Selling and Fulfillment Foundation base URL is invoked.

Creating an HTML Template

A template HTML enables rendering the look and feel of the mobile device screen. A template HTML defines which fields are included and how they are laid out. Each screen requires a template HTML file. The template HTML uses some custom mobile device tags in addition to the standard HTML tags.

The template HTML must adhere to the XSD defined in the `INSTALL_DIR/repository/xapi/template/merged/mobilescreens/rf.xsd` style sheet.

All template HTML files must reside in the `INSTALL_DIR/extensions/global/template/mobilescreens/uientity` directory. The name `uientity` refers to the screen entity resource that needs to be created for a mobile transaction.

The template HTML files for the inventory inquiry scenario are available for you to copy and reuse from the `INSTALL_DIR/xapidocs/code_examples/rfinventory/` directory.

Chapter 5. Creating JSP Files

JSP File Structure

The JSP files call the appropriate API (if needed), pick up the appropriate template XML, and pass the values returned by the API as data to the template XML.

A separate JSP file must be written for each of the following screen components:

- Screen - for invoking the HTML template and rendering the screen on the mobile client.
- Validation - for performing field level validations.
- Command button - for performing actions on clicking of the command button.

A JSP file for a mobile UI screen typically contains three sections, but not all bullets apply to every mobile JSP file:

Section	Function
1	<ul style="list-style-type: none">• Extracts values from the pageContext or Session.• Performs setAttribute for some of the input bindings.
2	<ul style="list-style-type: none">• Calls an API using callAPI.• Processes logic for the API output.
3	<ul style="list-style-type: none">• Sends Form or Forward Page• Sends Error

The example JSP files for the inventory inquiry scenario are available for you to copy and reuse from the *INSTALL_DIR*/xapidocs/code_examples/rfinventory/ directory.

JSP File Name and Directory Guidelines

When naming JSP files for mobile devices, use the following rules:

- The starting JSP file can have any name but the same name must be defined while adding the inner panel for the screen entity.
- The validation JSP file name syntax must be formName + "Val" + fieldname + ".jsp" format (for example, frmSearchValtxtItemId.jsp is invoked for validating the txtItemId field on the tab out of the txtItemId field in the frmSearch form).
- The name of the JSP being called on the click of a button must be named after the value of the action property in the URL. For example, if button has the URL value of /console/rfinventory.ppc?action=frmSearchUpdCmdInquire, then the JSP being invoked is named as frmSearchUpdCmdInquire.jsp.
- All JSPs must be added to the *INSTALL_DIR*/extensions/global/webpages/rf/wms/*uidentity* directory (for example, *INSTALL_DIR*/extensions/global/webpages/rf/wms/inventory/frmInventory.jsp).
- These JSPs use common utility methods as defined in the *INSTALL_DIR*/repository/eardata/platform/war/yfc/rfutil.jspf file.

Chapter 6. Style Reference for Mobile Device Interfaces

HTML Tags for Mobile Device Interfaces

The mobile UI uses the following HTML tags:

Tag	Valid Values/Detail
type	Valid values are: text, hidden, or button.
subtype	<p>Text type tags use the following values:</p> <ul style="list-style-type: none">• Label - Static text.• ProtectedText - Non-editable input.• Text - Input text box. <p>Hidden type tags use the value Hidden.</p> <p>Button type tags use the following values:</p> <ul style="list-style-type: none">• Command - HTML button.• CommandLogout - Logs the user out and displays the login prompt.• CommandBack - Switches to the previous view.• CommandNextView - Switches to the next view.
name	Name of the field. Cannot contain spaces.
value	Value of the field (Internationalized string from the resource bundle)
size	Length of the field.
maxlen	Maximum length of the field.
row	Row in which the field should appear.
col	Column in which the field should start.
validate	<p>Input data validated by the server. Valid values are:</p> <ul style="list-style-type: none">• Always - validate input data in all cases.• True - validate input data only if the old value is different from the new value.• False - do not validate input data.
mandatory	<p>Field usage validated by the server. Valid values are:</p> <ul style="list-style-type: none">• True - the user is required to specify a value for the field.• False - the user is not required to specify a value for the field.
inputbinding	XML binding for a field. The UI infrastructure resolves this binding and displays the value on the UI.
outputbinding	XML binding for a field when the field value passes to the next screen.
tag	Binding for the field recognized by the UI infrastructure for Pocket PC and WinCE applications. The purpose of this is same as the outputbinding. The tag syntax is "binding=x", where 'x' is an XML binding (similar to the ones in Console screens).
defaultoutput	If this is not set to False, when inputbinding is resolved as void, the UI infrastructure resolves outputbinding of a field, and displays its value in the UI.

Tag	Valid Values/Detail
url	<p>Specified only for "button/Command" field type/subtype. If set, the request is forwarded to the value of this attribute. It should always be of the form</p> <p>target + "?action=" + calledFormName</p> <p>where target is the value of the attribute target in the "form" element of the HTML while calledFormName is the name of the JSP to be invoked without the ".jsp" extension.</p>

Note: The sequence in which the UI infrastructure resolves an input tag is as follows:

1. Resolves inputbinding.
2. Resolves outputbinding if defaultoutput is not set to False.
3. If no resolution is found, uses the value attribute provided in the HTML.
4. If no resolution is found, uses the defaultvalue attribute provided in the HTML.
5. Resolves defaultbinding.

JSP Tag Library

Mobile UI screens for this application use the same JSP tags listed in the Sterling Selling and Fulfillment Foundation: *Customizing Console JSP Interface for End User Guide*.

Data Type Reference

The mobile UI for this application uses the same data types listed in the Sterling Selling and Fulfillment Foundation *Customizing Console JSP Interface for End User Guide*.

Chapter 7. Programming Standards for Mobile Device Interfaces

JSP File Standards for Mobile Device Interfaces

Although HTML code is embedded in Java Server Pages, strive to write JSP code that is easily readable. If you require some special XML manipulation that cannot be incorporated in the APIs, include a separate JSP file, so that HTML tags and Java code do not become mixed together.

Use the following standards when writing JSP files:

- Tab spacing - Set the editor tab spacing to 4.
- JavaScript files - Do not include any JavaScript in the JSP file. Put all JavaScript into a separate JS file.
- HTML tags - Type all HTML tags and attributes in lowercase letters.
- HTML attributes - Enclose all HTML element attribute values in double quotes. Single quotes and no quotes may work, but the standard is to use double quotes.
- HTML tables - Strictly adheres to XSD defined.
- Tags - Close all tags, whether required or not.
- Comments - Enclose all comments in the following manner: `<%/*.....*/%>`

Tip: When finished coding a form, open it in any visual HTML editor to validate that the HTML is well-formed.

Internationalization Standards for Mobile Device Interfaces

The Presentation Framework enables you to write an internationalized application by providing the following features that can be customized to be locale-specific:

- `i18n` JSP tag for literals
- Server-side error messages

Validating HTML Files for Mobile Device Interfaces

Validate your HTML files. You can use any commercial software package or free online application such as the World Wide Web Consortium (W3C) HTML Validator at <http://validator.w3.org/>. As an alternative, when you finish coding a form, you can open it in any visual HTML editor to validate that the HTML is well-formed.

Additionally, validate the HTML files against the XSD files.

Chapter 8. JSP Functions for Mobile Device Interfaces

JSP Functions Used for Mobile Device Interfaces

You can extend a mobile device interface by including any of the functions listed in the *INSTALL_DIR/repository/eardata/platform/war/yfc/rfutil.jspf* file in your JSP file. In alphabetical order, these functions are:

- addToTempQ
- clearTempQ
- deleteAllFromTempQ
- deleteFromTempQ
- getErrorXML
- getField
- getForm
- getStoredElement
- getTempQ
- getTempQValue
- replaceInTempQ
- resetAttribute
- sendForm

addToTempQ

Description

This mobile device JSP function adds the *keyName* and *keyValue* pair to TempQ in order to persistence data across JSPs. The TempQ utilities store name/value pair information on one page in the session and provide methods for accessing them on the subsequent screens.

This function also enables support of multiple duplicate key names.

Syntax

```
public void addToTempQ (String keyName, String keyValue, boolean allowDuplicates) throws Exception
```

```
public void addToTempQ (String keyName, String keyValue, Map m, boolean allowDuplicates) throws Exception
```

Input Parameters

keyName - Required. Name of the key to be stored in the TempQ.

keyValue - Required. Value of the key to be stored in the TempQ.

allowDuplicates - Required. Determines whether or not duplicate objects are allowed to be added to the TempQ.

m - Optional. Enables you to provide a map of name/value pairs in a `java.util.map`.

Example

The following example shows how the `addToTempQ` function can be used when multiple cases have to be scanned and multiple CaseIDs have to be stored in a TempQ:

```
addToTempQ("Case", "Case", caseMap, true);
```

clearTempQ

Description

This mobile device JSP function clears the TempQ. This is the first function to invoke before persisting any information in the TempQ. The TempQ utilities store name/value pair information on one page in the session and provide methods for accessing them on the subsequent screens.

Syntax

```
public void clearTempQ() throws Exception
```

Input Parameters

None.

Example

The following example shows how this function can be used to clear the TempQ:

```
clearTempQ();
```

deleteAllFromTempQ

Description

This mobile device JSP function deletes entries from TempQ for a given `keyName`. Use this after an exception to clear CaseIDs stored so far.

The TempQ utilities store name/value pair information on one page in the session and provide methods for accessing them on the subsequent screens.

Syntax

```
public void deleteAllFromTempQ(String keyName) throws Exception
```

Input Parameters

keyName - Required. This is the key for which TempQ entries are deleted.

Example

The following example shows how the `deleteAllFromTempQ` function can be used to remove all TempQ entries that correspond with the `keyName` `CaseScanned`:

```
deleteAllFromTempQ("CaseScanned");
```

deleteFromTempQ

Description

This mobile device JSP function deletes the TempQ entry for a given keyName and keyValue pair.

The TempQ utilities store name/value pair information on one page in the session and provide methods for accessing them on the subsequent screens.

Syntax

```
public void deleteFromTempQ(String keyName, String keyValue) throws Exception
```

Input Parameters

keyName - Required. Name of the key in the TempQ that requires deletion.

keyValue - Required. Value of the key in the TempQ that requires deletion.

Example

The following example shows how the getLocale function can be used in conjunction with the getDoubleFromLocalizedString function:

```
deleteFromTempQ("Case", "Case");
```

getErrorXML

Description

This mobile device JSP function returns an XML representation of error. The mobile device interprets this XML and renders an error page.

Syntax

```
public String getErrorXML(String error, String errorField)
```

```
public String getErrorXML(String error, String errorField, String severity)
```

Input Parameters

error - Required. Error description for the error to be shown to the user. This is usually derived by invoking the checkForError() function.

errorField - Required. Form field where the focus must be transferred to on clearing the error page.

severity - Optional. Displays the degree of error present. Recommended values in ascending order are: info, error, warning.

Example

The following example shows how this function can be used to get the error XML for rendering an error message on a mobile UI screen:

```
errorXML=getErrorXML(errorDesc, errorfield)
```

getField

Description

This mobile device JSP function is used in conjunction with the `getForm()` function.

Syntax

```
public YFCElement getField(YFCDocument formDoc, String fieldName) throws  
Exception
```

Input Parameters

formDoc - Required. Name of the YFCDocument from which the element must be extracted.

fieldName - Required. Name of the form field for which the other attributes need to be set.

Example

The following example shows the `getField` function.

Consider a form with `formName` as `formName`. The following code creates a YFCDocument from the XHTML form:

```
YFCDocument ydoc=getForm(formName);
```

Prior to setting the attributes of the form for a specific element, `getField` can be called as:

```
YFCElement dropoffLocationElem = getField(ydoc,"lblDropoffLocation");
```

To set the type and subtype attributes for the `dropoffLocationElem`, use:

```
dropoffLocationElem.setAttribute("type","hidden");  
dropoffLocationElem.setAttribute("subtype","Hidden");
```

getForm

Description

This mobile device JSP function reads the XHTML form for a given form name and returns a YFCDocument. The `currentEntity` name is prefixed to the formname and `.html` is suffixed. It looks for the file in the `INSTALL_DIR/repository/xapi/template/merged/mobilescreens/` directory.

The `getForm()` function is always used in conjunction with the `getField()` function.

Syntax

```
public YFCDocument getForm(String formName) throws Exception
```

Input Parameters

formName - Required. Name of the XHTML form.

Example

The following example shows how this function can be used to return a YFCDocument for the form "formName":

```
YFCDocument ydoc=getForm()
```

getStoredElement

Description

This mobile device JSP function returns the XML Element for all TempQ entries. Each keyName and keyValue entry can be obtained by traversing the Element.

Syntax

```
private YFCElement getStoredElement(YFCDocument ydoc, String keyName, String  
keyValue) throws Exception
```

Input Parameters

ydoc - Required. YFCDocument representation of the XHTML form.

keyName - Required. Name of the key in the TempQ.

keyValue - Required. Value of the key in the TempQ.

Example

The following example shows the getStoredElement function:

```
YFCElement criteria = getStoredElement(getTempQ(),"Criteria", "criteria");
```

getTempQ

Description

This mobile device JSP function retrieves the TempQ document object from Session. If a TempQ document does not exist, this function creates one. Since the return type of this function is YFCDocument, it is used to get a handle to the TempQ and thereafter is used to get its elements or attributes for some of its elements.

The TempQ utilities store name/value pair information on one page in the session and provide methods for accessing them on the subsequent screens.

Syntax

```
public YFCDocument getTempQ() throws Exception
```

Input Parameters

None.

Example

The following example shows how this function can be used to first get the TempQ documents and then later to get the node list for elements with the tag name LPN:

```
YFCNodeList lpnlist=()(getTempQ()), getElementsByTagName("LPN"));
```

getTempQValue

Description

This mobile device JSP function returns the value of the TempQ for a specific key. The TempQ utilities store name/value pair information on one page in the session and provide methods for accessing them on the subsequent screens.

Syntax

```
private String getTempQValue(String keyName) throws Exception
```

Input Parameters

keyName - Required. Name of the key for which the TempQ value is to be returned.

Example

The following example shows how the getTempQValue function can be used to return the keyValue corresponding to the CaseScanned key from the TempQ:

```
getTempQValue("CaseScanned");
```

replaceInTempQ

Description

This mobile device JSP function replaces the value in TempQ for a given key. The TempQ utilities store name/value pair information on one page in the session and provide methods for accessing them on the subsequent screens.

Syntax

```
public void replaceInTempQ(String keyName, String keyValue) throws Exception
```

```
public void replaceInTempQ(String keyName, String keyValue, String  
newKeyValue) throws Exception
```

```
public void replaceInTempQ(String keyName, String keyValue, Map m) throws  
Exception
```

Input Parameters

keyName - Required. Name of the key in the TempQ that is being replaced.

keyValue - Required. Value of the key in the TempQ that is being replaced.

newKeyValue - Optional. The new value of the key in TempQ that replaces the old value.

m - Optional. Map that should replace the existing keyValue.

Example

The following example shows how this function can be used to change the value of RecordCount from "1" to "resultMap":

```
replaceInTempQ("RecordCountResult","1",resultMap);
```

resetAttribute

Description

This mobile device JSP function removes the named attribute from request and PageContext.

Note: It is a good coding practice to reset an attribute before using it in the code.

Syntax

```
public void resetAttribute(String name)
```

```
public void resetAttribute(String name, Object value)
```

Input Parameters

name - Required. The name of the request attribute that needs to be reset.

value - Optional. The value of the request attribute that the attribute should be reset to.

Example

The following example shows how the resetAttribute JSP function removes the named attribute from request and PageContext:

```
resetAttribute("TaskList","");
```

sendForm

Description

This mobile device JSP function posts an HTML form and provides focus on a specific field on a subsequent JSP form. Three versions of syntax enable you to customize how data should display.

Syntax

```
public String sendForm(String formName, String focusField) throws Exception
```

```
public String sendForm(String formName, String focusField, boolean sendData)  
throws Exception
```

```
public String sendForm(YFCDocument formDoc, String focusField, boolean  
sendData) throws Exception
```

Input Parameters

formDoc - Required. Either formName or formDoc must be provided.

formName - Required. Either formName or formDoc must be provided.

focusField - Required. Form field where the focus must be transferred to in the invoked JSP.

sendData - Optional. Valid values: true and false.

Example

The following example shows how the sendForm function can be used so that the form corresponding to the YFCDocument ydoc is posted. On invocation of the subsequent JSP, the focus is transferred to the txtLocationId field and data is posted.

```
sendForm(ydoc, "txtLocationId", true)
```

Chapter 9. Configuring the Mobile Application

Configuring the Mobile Application

Sterling Selling and Fulfillment Foundation provide the ability to configure mobile screen parameters to meet specific implementation requirements.

Each menu on the Mobile Application screen has a Resource ID associated with it. You can create a new resource and associate it with the menu. For more information about creating a new resource and adding a menu entry, see "Creating Resources in the Applications Manager" and "Adding a Menu Entry" sections.

You can create an XML file and use it to configure the Mobile Application screens that has the same Resource ID, at both field level and form level. The following describes the format of a sample XML file.

```
<MobileConsoleOverride>
  <ResourceOverrides>
    <ResourceOverride ResourceId="ordrfreceive">
      <FormOverrides>
        <FormOverride name="frmReceiveIntoLocation">
          <Fields>
            <Field name="txtLocationId"
defaultvalue="DOCKLOCATION" validate="always"/>
          </Fields>
        </FormOverride>
        <FormOverride name="frmReceivingCriteria"
defaultfocusfield="txtShipmentNo" overridefocusfield="txtPONo">
          <Fields>
            <Field name="lblShipRef"
subtype="Hidden" />
          </Fields>
        </FormOverride>
      </FormOverrides>
    </ResourceOverride>
    <ResourceOverride ResourceId="fsrfputaway">
      <FormOverrides>
        <FormOverride name="frmPutawayEquipment">
          <Fields>
            <Field name="txtEquipmentId"
mandatory="true" />
          </Fields>
        </FormOverride>
        <FormOverride name="frmSKUPickInstruction">
          <Fields>
            <Field name="txtItemId"
defaultbinding="xml:/TaskList/Task/Inventory/@ItemId" validate="always"/>
          </Fields>
        </FormOverride>
      </FormOverrides>
    </ResourceOverride>
    <ResourceOverride ResourceId="nerfmanualmove">
      <FormOverrides>
        <FormOverride name="frmEquipment"
onloadaction="F3" />
        <FormOverride name="frmSource"
defaultfocusfield="txtLocationId" overridefocusfield="txtInv">
          <Fields>
            <Field name="txtInv"
executeaction="F8"/>
          </Fields>
        </FormOverride>
      </FormOverrides>
    </ResourceOverride>
  </ResourceOverrides>
</MobileConsoleOverride>
```

```
        </FormOverride>
    </FormOverrides>
</ResourceOverride>
</ResourceOverrides>
</MobileConsoleOverride>
```

Configuring the Mobile Application Screens

About this task

This section explains how to configure Mobile Application screens.

To configure a Mobile Application screen:

Procedure

1. Identify the Resource ID and Form Name by pressing the F1 key.
2. In `<INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/<ResourceId>` folder, identify the HTML file for the Form Name.
3. Identify the fields you want to override.
4. Create an XML file in the `<INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/overrides` folder to configure the fields by changing the value of the attributes.
5. Run `<INSTALL_DIR>/bin/deployer.sh -t resourcejar`. This puts the xmls that are in the `<INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/overrides` folder to `resources.jar`.
6. Rebuild and redeploy the EAR.

New Attributes

Sterling Selling and Fulfillment Foundation have introduced the following new attributes to the field element:

- mandatory

You can mandate a field by setting the value of the attribute to "true" in the XML file. When you configure a field as mandatory, the functional keys, other than F1, F4, F7, and F10 do not let you proceed further.

For example, you can mandate scanning the equipment number when performing the pick process.

Note: The error message "Entry is mandatory" is thrown if you try to skip a field whose mandatory attribute is set to "true".

- defaultvalue

You can default the value of a field on the screen by configuring this attribute at the field level in the XML file.

For example, if a warehouse has only one location where it receives inventory, the Location ID can be defaulted so that the user do not have to scan or enter this field every time when receiving inventory.

Note: When configuring the "defaultvalue" attribute, if the "validate" attribute in that field is set to "true", you must ensure that it is set to "always".

- defaultbinding

You can default the value of a field with the value of some other field by assigning the "defaultbinding" to the "inputbinding" attribute of the other field.

For example, in the Pick screen, you can default the item ID field with the suggested item ID by setting the "defaultbinding" attribute of the item ID field to the "inputbinding" attribute of the suggested item ID.

Note: When configuring the "defaultbinding" attribute, if the "validate" attribute in that field is set to "true", you must ensure that it is set to "always".

- executeaction

You can press the Tab key and emulate the action of a button available on the screen without pressing the button or using the corresponding functional key by configuring this attribute in the XML file.

For example, you can emulate the F3 action without pressing this key or choosing Go by setting the value of this attribute to "F3".

- onloadaction

You can skip a screen and move to the next one using an action available on the current screen by configuring this attribute at the form level.

For example, you can skip the Equipment Entry screen, where you are asked to enter or scan the Equipment #, and move to the next screen without choosing the Go button or pressing F3 by setting the value of this attribute to "F3".

- defaultfocusfield, overridefocusfield

You can default the cursor to point to a particular field from the default field by configuring these attributes at the form level.

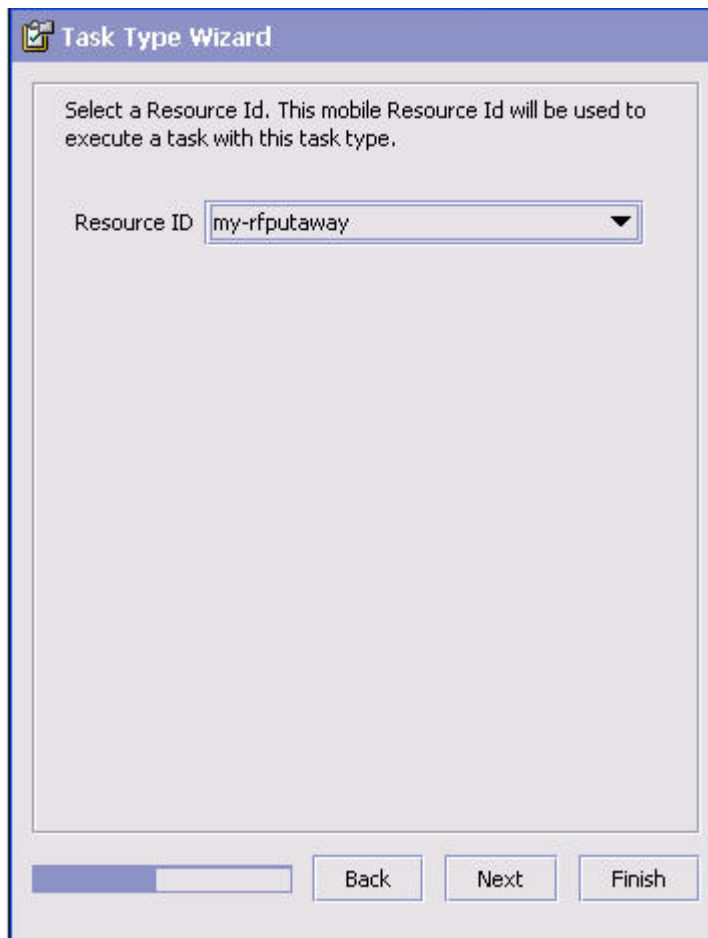
For example, when performing ad hoc move, in the pick screen, you can default the cursor to point to the Item/Case/Pallet field instead of Location/Pallet/Case field by setting the value of "defaultfocusfield" to "txtLocationId" and "overridefocusfield" to "txtInv".

Note: You cannot configure the following screens for a single menu item. Any modifications made to these screens are not for a specific menu item, Therefore, it is visible in any pick-related screens. All these screens belong to the rftask resourceId.

- frmTaskCriteria
- frmRetrievalCriteria
- frmPartialDepositInstruction
- frmSKUPickingInstructions
- frmDepositException
- frmPickException
- frmTaskExpirationDate
- frmTaskLotEntry
- frmTaskSerialEntry
- frmTaskSecondarySerialEntry
- frmConfirmEmptyLocation
- frmWaitingForTasks
- frmNoOpenTasks
- frmTaskInvalidTag

Associating Resource ID with a Process

Users may want to define different screen behaviors for any task-based process followed in the implementation. In such situations, users can create a new Resource ID and associate it with the task type in the Task Type wizard.



The Mobile Application loads the Resource ID specified in the task type wizard when performing tasks of this type. The user can configure the UI for this Resource ID and get a different screen behavior.

Use Cases

Pick and Deposit **About this task**

Requirement — When picking a pallet, scan the pallet LPN and go to the Deposit screen. In the Deposit screen, the LPN that you scanned at the time of picking is defaulted and the system asks you to scan the deposit location.

Current[®] behavior — To pick and deposit a pallet, scan the pallet LPN at the time of picking and click the Deposit button or press F8 key to go to the Deposit screen. In the Deposit screen, you need to scan the suggested LPN and suggested deposit location. Click the Enter button or Go button or press the F3 key to deposit the pallet.

Configurations

To configure the Pick screen to meet the above requirement:

Procedure

1. From the Menu page, select Putaway and press Enter.
2. In the Equipment Entry screen, scan the Equipment # and press Enter.
3. In the Case Pick screen, press F1 to identify the Resource ID ("rfputaway") and Form Name ("frmCasePickInstruction").
4. From the <INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/rfputaway folder, identify the form by selecting the frmCasePickInstruction.html file.
5. Identify the field you want to override ("txtCaseId").
6. Identify the field attributes to overrides and create a field level override XML file in the following format.

```
<FormOverride name="frmCasePickInstruction">
  <Fields>
    <Field name="txtCaseId" executeaction="F8"/>
  </Fields>
</FormOverrides>
```

With this configuration, after you scan an LPN, the Deposit button or F8 action invokes.

Note: If you scan an invalid data in the text field, an error is thrown and the Deposit button or F8 action does not invoke.

7. In the Deposit screen, press F1 to identify the Resource ID ("rfputaway") and Form Name ("frmCaseDepositInstruction").
8. From the <INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/rfputaway folder, identify the form by selecting the frmCaseDepositInstruction.html file.
9. Identify the fields you want to override ("txtCaseId" and "txtLocationId")
10. Identify the field ("txtSuggestedCaseId") whose value you want to default in the txtCaseId field.
11. Copy the value of inputbinding ("xml:/DepositLocation/@LabelAll").
12. Use the mobileConsoleOverrides.xml file and write an element in the following format:

```
<FormOverride Name="frmCaseDepositInstruction">
  <Fields>
    <Field Name="txtCaseId"
      defaultbinding="xml:/DepositLocation/@LabelAll"/>
      validate="always" />
    </Field>
    <Field Name="txtLocationId"
      executeaction="F3" /> v
      alidate="always"/>
    </Fields>
</FormOverride>.
```

Note: Since the txtCaseId and txtLocationId fields have "validate" set to "true", you must set the value "always" in mobileConsoleOverrides.xml.

With this configuration, whatever Case ID is suggested in the Deposit screen is defaulted in the text field. Whenever, you scan a valid location, the Deposit button or F3 action invokes.

13. Create an override XML file with the above elements in the <INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/overrides folder. The XML file is in the following format:

```

<MobileConsoleOverride>
  <ResourceOverrides>
    <ResourceOverride ResourceId="rfputaway" >
      <FormOverrides>
        <FormOverride Name="frmCasePickInstruction">
          <Fields>
            <Field Name="txtCaseId" executeaction="F8"/>
          </Field>
        </Fields>
      </FormOverride>
      <FormOverride Name="frmCaseDepositInstruction">
        <Fields>
          <Field Name="txtCaseId"
            defaultbinding="xml:/DepositLocation/@LabelAll"/>
            validate="always"/>
          </Field>
          <Field Name="txtLocationId"
            executeaction="F3"/>
          </Field>
        </Fields>
      </FormOverride>
    </FormOverrides>
  </ResourceOverride>
</ResourceOverrides>
</MobileConsoleOverride>

```

Defaulting Dock Location

About this task

Requirement - While performing receiving, the location is defaulted in the Receive screen and you are not asked to scan it every time you perform a receiving task.

Current Behavior — The system asks you to scan the location where you want to receive the shipment after you scan or enter the receiving criteria.

Configurations

To configure the Receive screen to meet the above requirement:

Procedure

1. From the Menu page, select Receipt and press Enter.
2. In the Receive screen, select Purchase Order as document type and press Enter.
3. Press F1 to identify the Resource ID ("rfreceive") and Form Name ("frmReceiveIntoLocation").
4. From the <INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/rfreceive folder, identify the form by selecting the frmReceiveIntoLocation.html file.
5. Identify the field you want to override ("txtLocationId").
6. Use the override XML file located in the <INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/overrides folder and write an element in the following format:

```

<MobileConsoleOverride>
  <ResourceOverrides>
    <ResourceOverride ResourceId="rfreceive" >
      <FormOverrides>
        <FormOverride Name="frmReceiveIntoLocation">
          <Fields>
            <Field Name="txtLocationId"
              defaultvalue="DOCK-LOCATION" validate="always"/>
          </Fields>
        </FormOverride>
      </FormOverrides>
    </ResourceOverride>
  </ResourceOverrides>
</MobileConsoleOverride>

```

```

        </FormOverride>
    </FormOverrides>
</ResourceOverride>
</ResourceOverrides>
</MobileConsoleOverride>

```

With this configuration, the location defaults in the Receive screen.

Skipping a Screen

About this task

Requirement - When performing an ad hoc move, you want to skip the Equipment Entry screen and go directly to the Source screen without clicking the Go button or pressing the F3 key.

Current Behavior - After selecting Ad hoc Move, the system asks you to scan the Equipment # after which you click the Go button or press the F3 key to go to the Source screen.

Configurations

To configure the Ad Hoc Move screen to meet the above requirement:

Procedure

1. From the Menu page, select Ad hoc Move and press Enter.
2. Press F1 to identify the Resource ID ("rfmanualmove"), Form Name ("frmEquipment") and the action you want to emulate (F3).
3. Use the override XML file located in the `<INSTALL_DIR>/repository/xapi/template/merged/mobilescreens/overrides` folder and write an element in the following format:

```

</MobileConsoleOverride>
</ResourceOverrides>
  <ResourceOverride ResourceId="rfmanualmove" >
    <FormOverrides>
      <FormOverride name="frmEquipment" onloadaction="F3"/>
    </FormOverrides>
  </ResourceOverride>
</ResourceOverrides>
</MobileConsoleOverride>

```

With this configuration, you go directly to the Source screen after selecting Ad hoc Move from the menu.

Chapter 10. Menu-Level Customization of the Mobile Application

Menu-Level Customization of the Mobile Application

There are occasions when a warehouse needs to set up different menu options for the Mobile Application to specify resource-level parameters for different behaviors. For example, a new Pack Shipment menu is created to enable automatic recording of item details in the container being packed or a new Ad hoc move menu is created for a particular activity code and activity group.

When customizing the Mobile Application, the JSPs and HTMLs are mostly changed. Therefore, a new resource ID for the new customized version is created. All the JSP and HTML files corresponding to the factory shipped resource IDs are copied into new directories corresponding to the new resource ID. These newly created directories are at the same level as the old directories from which the JSP and HTML files are copied. The HTML files are modified by replacing the occurrence of <resourceId>.ppc with <new resourceId>.ppc. The new resource ID is then associated with a new menu item.

Menu-Level Customization

When performing menu-level customizations, the JSP and HTML files are not modified. The Sterling Warehouse Management System enables you to create a new menu for the Mobile Application and associate it with a new resource ID, without making copies of the JSP and HTML files.

The following sections explain how to avoid copying the JSP and HTML files when performing menu-level customizations.

Avoiding Copying of HTML Files

About this task

To avoid copying of HTML files:

Procedure

1. Save the factory-shipped resource with a new ID by adding a prefix that ends with a hyphen (-) to the old resource ID.

For example, to create a new Ad hoc move, save the resource "manualmove" with a prefix "1-". The new resource ID is "1-manualmove".

2. Create a new menu and associate the new resource ID with it.

This ensures that the newly created resource also accesses the same HTML files accessed by the factory-shipped resource.

Avoiding Copying of JSP Files

About this task

To avoid copying of JSP files:

Procedure

1. Save the factory-shipped resource with a new ID.
2. In the resource details, the `JavaServerPage` parameter appears. Remove `"/extn"` from the `JavaServerPage` parameter.
3. Create a new menu and associate the new resource ID with it.
This ensures that the newly created resource also accesses the same JSP files accessed by the factory-shipped resource.

Chapter 11. Configuring Mobile Application User Interface Components

By configuring the properties in the YMAProperties.xml file, the look and feel of the Mobile Application can be changed.

Note: The configuration capabilities described in this section are valid only for Microsoft Windows CE and Pocket PC mobile terminals.

Device	Properties File
LXE MX7 handheld	YMAProperties.MX7.xml
LXE VX3X series truck mount	YMAProperties.VX3X.xml
PocketPC mobile terminal	YMAProperties.ppc.xml
Symbol VRC7900 series truck mount	YMAProperties.vrc7900.xml

The Mobile Application reads the configuration from the YMAProperties.xml file. Therefore, locate the property XML that matches your mobile device (listed in the above table), and rename it as YMAProperties.xml.

The properties listed in the following are provided out-of-the-box.

Property	Description
CharHeight	Height of a character in pixels.
CharWidth	Width of a character in pixels.
TextFontName	Specifies the font used in the text boxes.
TextFontSize	Specifies the size of the font used in the text boxes.
TextIsBold	Specifies whether the characters in the text box should appear as bold.
LabelFontName	Specifies the font used in the label fields.
LabelFontSize	Specifies the size of the font used in the label fields.
LabelHeight	Height of the label fields in pixels.
LabelsBold	Specifies if the characters in the label fields should appear as bold.
LineHeight	Height of a line in pixels.
MaxYValue	Specifies the height of the renderable area in a screen in pixels. When a character is requested to be positioned at a value that is higher than this, control automatically moves to a second column.
ColumnWidth	Specifies the number of pixels to advance to the right in order to move to the next column.
DeActivateOKButtonOnError	If the mobile device has an auto scan or auto click facility, it closes the error message dialog box automatically. This property allows the deactivation of the OK button in the error message dialog box so that the Mobile Application continues to display the error message to the user.

Property	Description
SoundForError	This property allows to specify a different beep in an error scenario. Identify a sound file (.wav) available on the mobile device, and set without extension (.wav) to this attribute.
enablePasteFunction	Allows paste facility in the text boxes.
sipEnabled	This property enables input from the soft input panel of a mobile device.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

1623-14, Shimotsuruma, Yamato-shi

Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be

incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© IBM 2011. Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2011.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium and the Ultrium Logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Connect Control Center[®], Connect:Direct[®], Connect:Enterprise[™], Gentran[®], Gentran[®]:Basic[®], Gentran:Control[®], Gentran:Director[®], Gentran:Plus[®], Gentran:Realtime[®], Gentran:Server[®], Gentran:Viewpoint[®], Sterling Commerce[™], Sterling Information Broker[®], and Sterling Integrator[®] are trademarks or registered trademarks of Sterling Commerce[™], Inc., an IBM Company.

Other company, product, and service names may be trademarks or service marks of others.



Product Number: xxxx-xxx

Printed in USA