IBM Case Manager 5.2: Customize and Extend the Features

Course Code: F217
ERC: 1.0
Volume Number: 1/1

Participant Guide

Template Version#: 11.0/1.0

Trademarks

IBM® and the IBM logo are registered trademarks of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

DB2®	HACMP™	System i™
System p™	System x™	System z™

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 Minister for the Cabinet Office, 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.

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

September 2014 edition

TABLE OF CONTENTS

Lesson 2.2: Create a Case Custom Workbench Page

Unit 3: Develop Custom Widgets

Lesson 3.1: Custom widget development overview

Lesson 3.2: Create catalog and widget definition files

Lesson 3.3: Implement a custom widget

Lesson 3.4: Build and register a widget package

Lesson 3.5: Create a widget with a toolbar and a menu

Lesson 3.6: Build and deploy a widget as an EAR file

Lesson 3.7: Update an existing package with new widgets

Course Overview

This Course Overview provides you with an outline of the course, its objectives, agenda, course organization and other details specific to this course.

Course Description

Overview

This course describes various options that IBM Case Manager provides for developing the user experience for your application.

Course Objectives

After completing this course, you should be able to:

- Customize the Case Manager Client User Interface
  - Customize the Banner and Login Page
  - Configure Icons and Labels
  - Specify Viewers for File Types
  - Create and Customize Help Topics
  - Customize Toolbar and Menu
- Use Script Adapter for customization.

Audience

The intended audiences for this course are:

- Developers who are responsible for
  - Customizing and extending the IBM Case Manager features.
  - Building a customized user interface and applications for IBM Case Manager.
- Anyone who needs to know the capabilities of IBM Case Manager.

Prerequisites

The prerequisites for taking this course are:

- F212 - IBM Case Manager 5.2: Build and Migrate a Solution
- F120 - IBM Content Navigator 2.0.2: Customize and Extend the Features
- Experience with Eclipse IDE for developing web applications.
- Intermediate level of expertise in the following technologies:
  - Java and JavaScript
  - JSON and Dojo (version 1.8.4)
  - HyperText Markup Language (HTML 5)
  - Cascading Style Sheets (CSS3)

Options for customizing IBM Case Manager Client

Options

IBM Case Manager provides various options to develop the user experience for your application. Each option requires different development skill set and time to achieve the goal.

- Customize the Case Client appearance
- Use Scripts and Script Adapter widget to customize Case Manager Client
- Develop Content Navigator plug-ins and custom widgets
- Implement the External Data Service (EDS) interface

The following sections describe each option and specify the unit names in this course that provides information and lab exercises for each option.

Customize the Case Client appearance

You can change the appearance of the Case Client in the admin tool without writing any code. This option is helpful to make quick changes to comply with the visual standards of an organization.

- Configure visual aspects for the Case Client banner and login page.
  - Specify the name for your solution
  - Add a company logo to both the login page and the banner.
  - Alter the colors that are used in the banner.
- Configure visual aspects for the icons and labels:
  - Change icons for specific mime-types.
  - Modify the labels that are used in the desktop.
- Add custom help topics.
- Extend the Toolbar and Menus.

Note

For more information, see “Unit 1- Customize the Case Manager Client User Interface” of this course.

Use Script Adapter widget to customize Case Manager Client

Note

For more information, see “Unit 2 - Use Script Adapter widget to customize Case Manager Client” of this course.

Implement the External Data Service (EDS) interface

The IBM FileNet P8 repositories that are used in your solution have specific options for how properties are defined in your object model.

EDS allows you to change behavior of property data fields dynamically during run time in a light-weight manner

EDS REST interface implementation enables the following tasks:

- Look up the choice list values for a property or dependent property
- Set minimum and maximum property values
- Implement property validation and error checking

Note

For more information, see “Unit 4 - Implement External Data Services” of this course.

Develop a widget

An IBM Content Navigator plug-in enables developers to add new functions, change existing behavior and appearance of the application or create new application. It is a powerful mechanism and the most flexible option for customizing user experience within your solution.

A plug-in consists of one or more extension points that can be split into two groups:

Required skills

- Java and JavaScript
- JSON and Dojo 1.8.4
- HTML5 and CSS3

Note

For more information, see “Unit 3 - Create Custom Widgets” of this course.

1

Customize the Case Manager Client User Interface

This unit provides guidance for customizing the IBM Case Manager Client user interface in the admin tool without having to write any code.

LESSON 1.1: Customize the Banner and Login Page

What this lesson is about?

This lesson describes how to customize the Case Manager Client appearance for the banner and login page.

What you should be able to do?

After completing this lesson, you should be able to:

- Customize the banner and logon page.

How you will check your progress?

- Hands on labs.

References

IBM Case Manager Version 5.2 Information Center

IBM Redbooks publication: Advanced Case Management with IBM Case Manager

Customize the Case Manager Client user interface

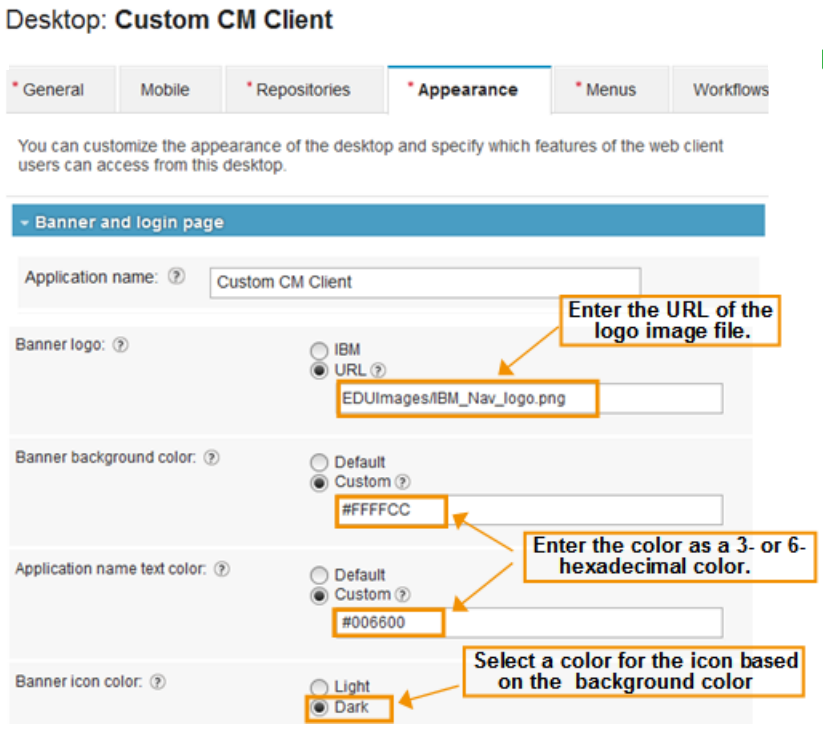
IBM Case Manager Client is based on IBM Content Navigator. Its visual characteristics are shown with a customized theme.

- The theme determines the overall appearance, such as fonts and colors.
- You can make the following changes in the administration tool without changing the theme:
  - Add your company logo to both the login page and banner of the client.
  - Add useful notes to the login page.
  - Alter the color of the desktop banner.
  - Change icons.
  - Alter the labels that are used in the client.

Customize the Banner appearance

Banner configuration

You can customize the banner without any custom code in the administration tool.



URL for logos

- A URL references the New logos.
  - If the image file is in the IBM Content Navigator web client application, enter a relative path.
  - Example: EDUImages/IBM_Nav_logo.png.
  - You can also create a web application and add your logos to the Web Content folder.
  - If you deploy it in a highly available configuration, enter a fully qualified path that is available to all instances of IBM Content Navigator.

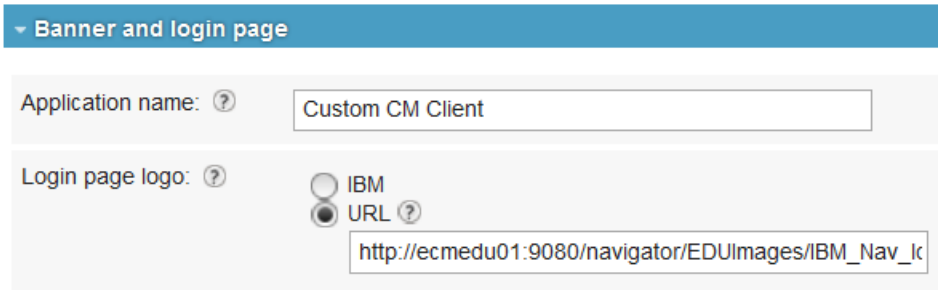
    Example: http://<server_name>/<application name> /<file name>
- The logo must be no larger than 72 pixels wide and 32 pixels high.
  - The IBM banner logo in IBM Content Navigator is 72 x 24 pixels.
- The logo image must have a transparent background

Customize the login page

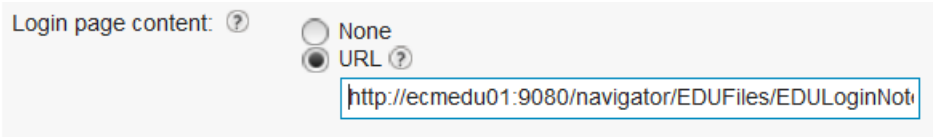
Login page configuration

You can customize the Login page logo and login page content in the administration tool.

- Enter the URL for the logo file.



- Enter the URL for the HTML file for the login page notes.



URL for login logo and login page content

- A URL references the new logo or the content file.
  - If the file is in the Content Navigator web client application, enter a relative path.
  - You can also create a web application and add your files to the Web Content folder.
- The size of the logo can be 150 x 112 pixels.
- The logo image must have a transparent background.

Login page content

- You can add content to the login page to provide information or the latest news to users when they access IBM Case Manager Case client.
- Example of information:
  - Guidance about the use of the system
  - Help desk contact procedures
  - Any planned downtime for system maintenance

- To add login page content, prepare an HTML file that contains the information that you want.
  - Add the file to a web application that is available to all systems where IBM Content Navigator is installed.

Demonstrations

Customize the Case Manager Case client Banner

Click here to watch the demonstration.

Customize the Case Manager Case client Login Page

Click here to watch the demonstration.

Exercise 1.1.1: Customize the banner

Introduction

In this lab, you make a copy of the existing Case Manager Case client and save it as another application. You customize the banner for your application.

User accounts

Type	User ID	Password
Operating system	administrator	passw0rd
IBM Case Manager Administrator	P8Admin	IBMFileNetP8

Passwords are always case-sensitive.

Procedures

Procedure 1: Start WebSphere Application Server

1. Click Start > All Programs > IBM WebSphere > IBM WebSphere Application Server V8.5 > Profiles > AppSrv01 > Start the server.
   - You can also use the `Start Server1.bat` file in the WebSphere Admin folder on the desktop.
2. Wait for the Start the server page to close.

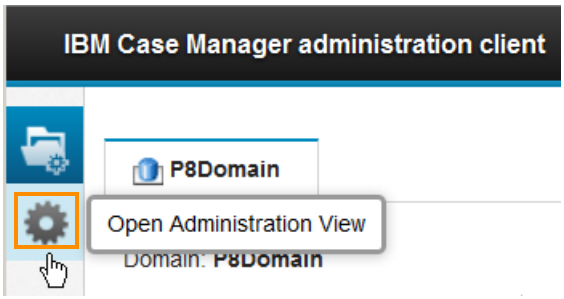
Note

For more information about “Start and stop System Components”, see the Appendix at the end of unit.

Procedure 2: Make a copy of the existing Case Manager client

1. Start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`

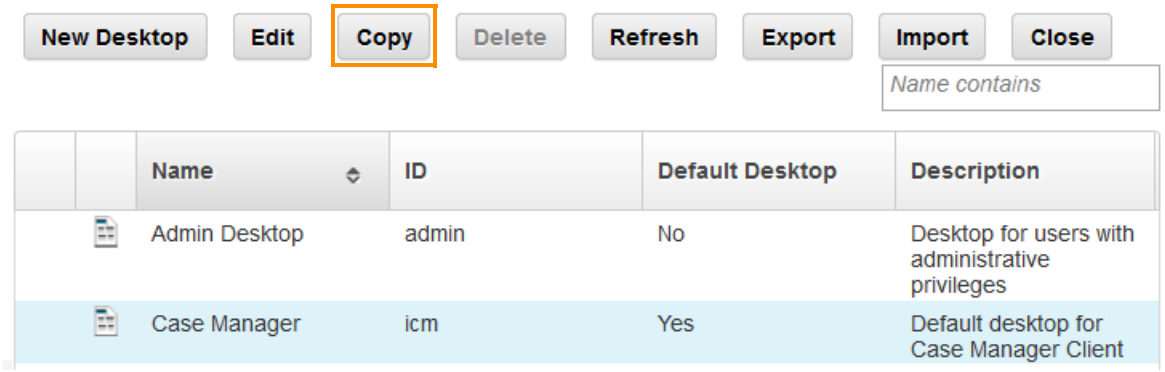
2. Copy the Case Manager Client application.
   a. On the administration client, click the “Open Administration View” icon.



Troubleshooting

If the Administration View icon is not visible, refresh the browser. You can also access the administration page with this URL: `http://ecmedu01:9080/navigator/?desktop=admin`

   b. The Desktops tab opens. In the Desktops tab, select Case Manager and click Copy.



3. In the New Desktop tab, enter a name for the desktop (Example: `Custom CM Client`).
   a. The ID field is automatically populated. Edit the ID value to custo

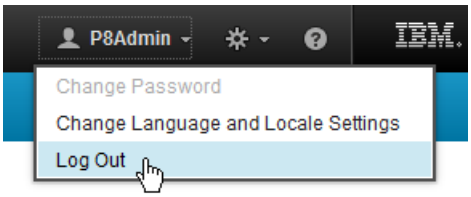
Note

The ID value is case-sensitive. You reference this ID in the URL for this application.

4. Click the Appearance subtab.
   a. Edit the Application name to `Custom CM Client`.
5. Click Save and Close.
6. Verify that the new application is listed in the Desktops tab.

7. Optional verification: Start the custom client application that you created.
   a. In a new browser tab, go to `http://ecmedu01:9080/navigator/?desktop=custom`
   b. Enter the login credentials.
      – User name: `P8admin`
      – Password: `IBMFileNetP8`

      The custom client application opens and the banner shows the new name.
8. Observe that the custom client application is identical to the default client and has the same `Cases` and `Work` tabs.
9. Log out of the custom client application and close the browser tab.



Procedure 3: Edit the new client to customize the banner

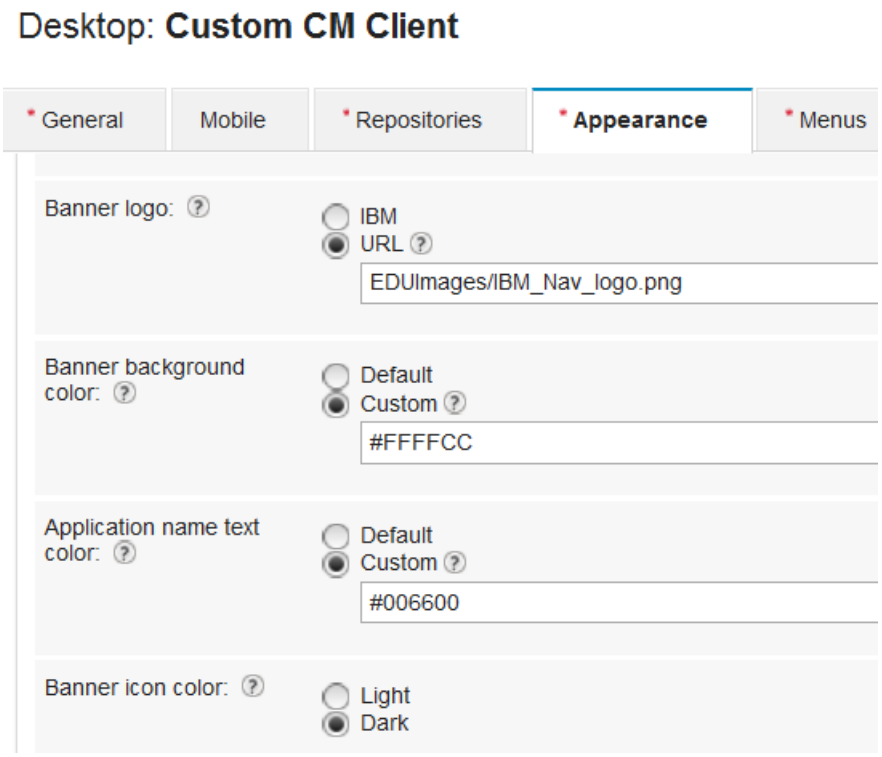
1. If it is not already opened, start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`

   a. On the administration client, click the “Open Administration View” icon on the left pane. If this icon is not visible, refresh the browser.
2. Edit the client.
   a. In the Desktops tab, right-click `Custom CM Client` that you created and click Edit.
   b. Click the Appearance subtab.
3. For the Banner and login page section, enter the values in the table.

Field Name	Option	Value
*Banner logo	URL	EDUImages/IBM_Nav_logo.png
Banner background color	Custom	#FFFFCC
Application name text color	Custom	#006600
Banner icon color	Dark	

*An image for the banner logo is already copied into the `C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\navigator.ear\navigator.war\EDUImages` folder.

The completed page looks similar to the one in the following screen capture.

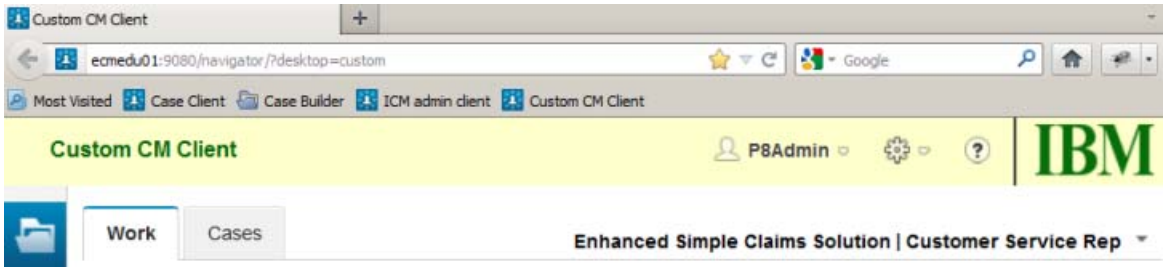


4. Click Save and Close.
   a. Log out of the administration page.
   b. Close the browser.

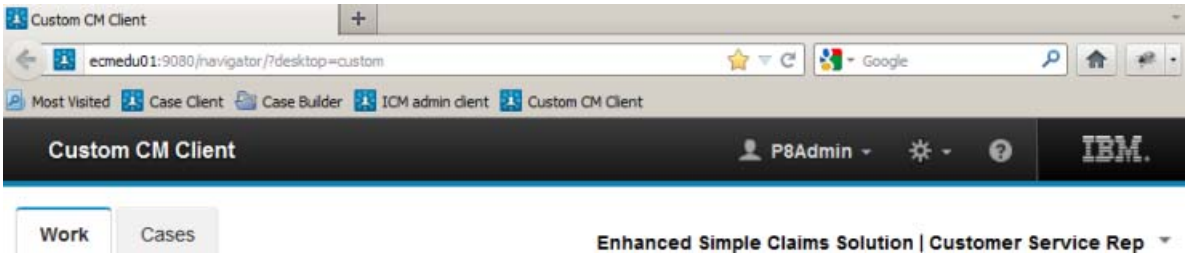
Procedure 4: Test the customized banner

1. Log in to the Custom CM Client in a new browser.
   - URL: `http://ecmedu01:9080/navigator/?desktop=custom`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Verify the following changes (as you configured) in the banner for the client.
   - The banner background color
   - Banner logo
   - Application name text color

After customization:



Before customization:



3. Log out of the custom client and close the browser.

Exercise 1.1.2: Customize the login page

Introduction

In this lab, you customize the login page for your application to add notes and a logo.

Procedures

Procedure 1: Edit the new client to customize the login page

Note

Two files that are required for this procedure are already copied to the application server directory.

1. Verify the following files:
   - An image file:

     `C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\navigator.ear\navigator.war\EDUImages\IBM_Nav_logo.png`
   - An html file:

     `C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\navigator.ear\navigator.war\EDUFiles\EDULoginNotes.html`
2. Start the IBM Case Manager administration tool.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
3. Edit the client:
   a. On the administration tool, click the “Open Administration View” (gear) icon on the left pane.
   b. In the Desktops tab, right-click Custom CM Client and click Edit.
   c. Click the Appearance subtab.
4. For the banner and login page section, enter the values in the following table

Field Name	Option	Value
Login page logo	URL	http://ecmedu01:9080/navigator/EDUImages/IBM_Nav_logo.png
Login page content	URL	http://ecmedu01:9080/navigator/EDUFiles/EDULoginNotes.html

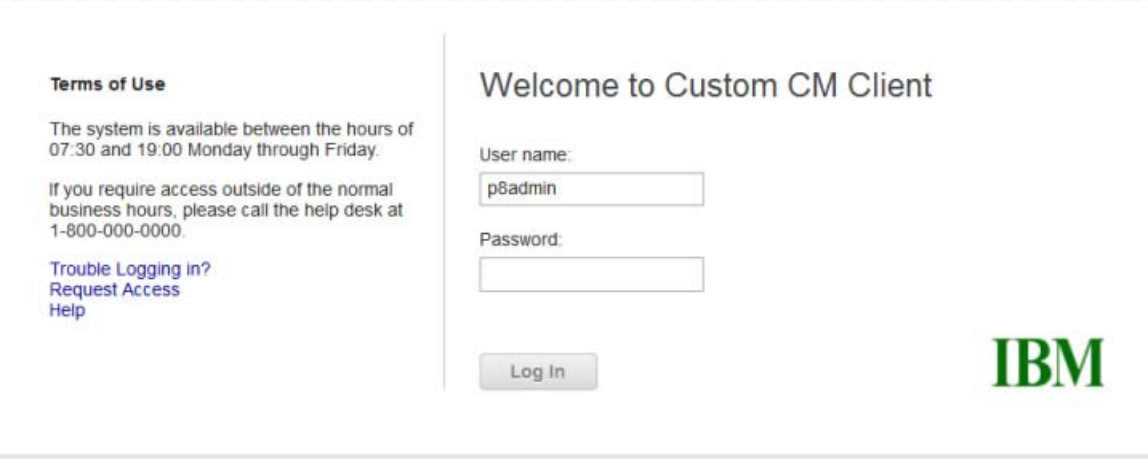
Note

You can also enter a relative path for the files (Example: `EDUFiles/EDULoginNotes.html`)because these files are copied to the same web application directory. If you are using a file from a different web application, then you must enter a fully qualified URL address.

5. Click Save and Close.
   a. If you are prompted, click Close in the Information window.
   b. Log out of the Administration page and close the browser.

Procedure 2: Test the customized login page

1. Log in to the Custom CM Client in a new browser.
   - URL: `http://ecmedu01:9080/navigator/?desktop=custom`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Test the following changes to the login page of the custom client.
   – A new section on the left that allows the user to get help with logging in to the application.
   – A new logo in the lower right corner.



3. Log out of the Custom Desktop and close the browser.

LESSON 1.2: Configure Icons and Labels

What this lesson is about?

This lesson describes how to customize the icons and labels for a custom Case Manager Case client.

What you should be able to do?

After completing this lesson, you should be able to:

- Change the icons.
- Modify labels.

How you will check your progress?

- Hands on labs.

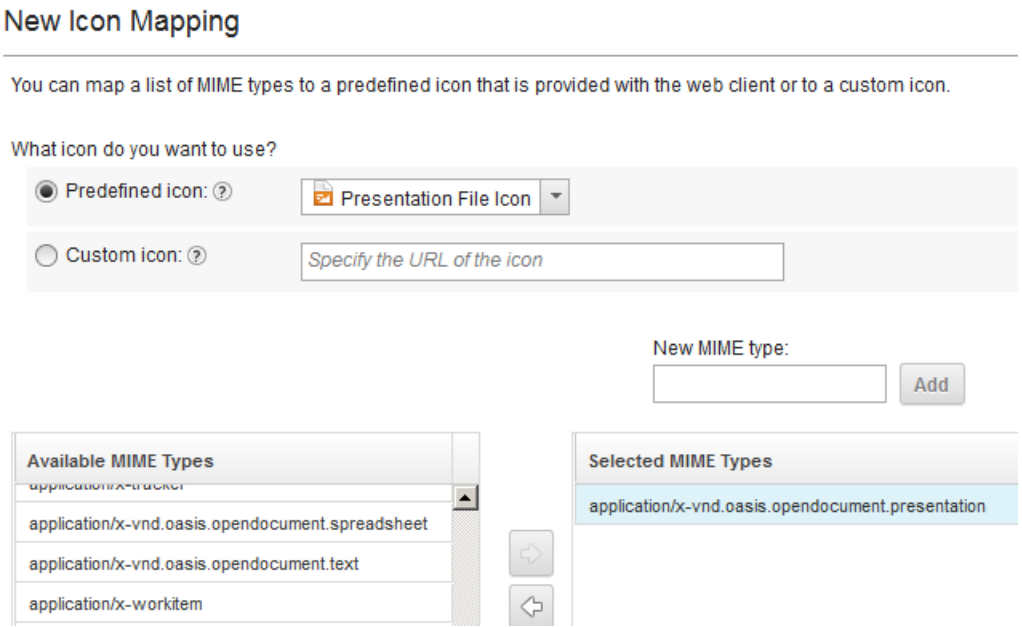
Change Icons for the client

Configure the icons that are shown in the web client

- You can change the icons that are displayed for document MIME types without any code.
  - MIME types indicate the type of document, such PDF or HTML.
  - The changes that you make apply to all of the clients in your environment.
- IBM Content Navigator provides predefined icons for MIME types.
  - You can map MIME types to the predefined icons.
  - You can also customize the icons to match icons that are used in your organization.
  - Example: Microsoft Office applications.

Create an Icon Mapping for MIME Types

You can create Icon Mappings to associate an icon to a MIME type.



- You can associate more than one icon for a MIME type.
  - If a MIME type has more than one Icon Mapping, the first icon in the list is used.

Edit an existing Icon Mapping for MIME Types

- You can edit an existing Icon Mapping to associate a new icon to a MIME type.
- In the administration tool for the web client, click Settings.
  - In the Settings tab, click the Icon Mapping subtab.
- Select an existing MIME type from the MIME Type Icons section.
  - Click Edit to map a MIME type to a predefined icon or to a custom icon.

Modify the labels in the Case Manager web client

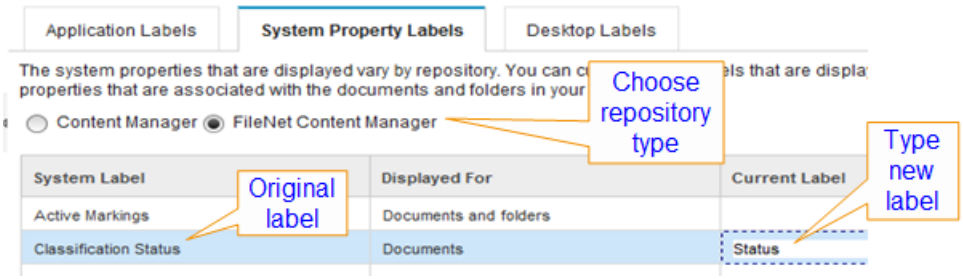
Customize labels

Customizing the labels are useful when users wants to use their own terminology.

- Changes that are made to Application Labels and System Property Labels apply to all the desktops in your environment.
- Each locale can have its own label text.
- The repositories use System Property Labels that show on desktops.

System Property Labels

The diagram shows how to change the default System Property Labels.



- Customize the labels in the Labels > System Property Labels tab.
- You can customize the labels for the system properties that are associated with the documents and folders in your repositories.
- Any changes that you make to the system property display names in the web client do not affect the property names and values that are configured on your repository.

Exercise 1.2.1: Customize an icon for Case Manager Client

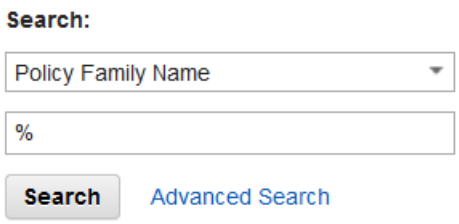
Introduction

In this lab exercise, you associate a custom icon for the text/plain and application/rtf document MIME types that are stored in the repository.

Procedures

Procedure 1: Check the default icon

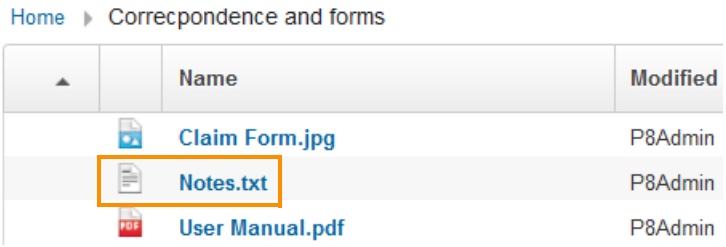
1. Start the Custom Case Manager Client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=custom`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Open a case.
   a. Click the Cases tab.
   b. In the Search section, select Policy Family Name from the list.
   c. Enter `%` in the second text box of the Search widget and click the Search button.



   d. Available cases are listed in the Case List widget.
3. Open the Case details for a case.
   a. Click the link for the case with `Smith` as the Policy Family Name value from Title column.
   b. The Case details are shown in a separate tab.
4. Open the documents list for this case.
   a. In Documents subtab, click Correspondence and forms folder.

b. To see the complete list of documents, collapse the Timeline Visualizer widget in the middle of the page by clicking the down arrow.

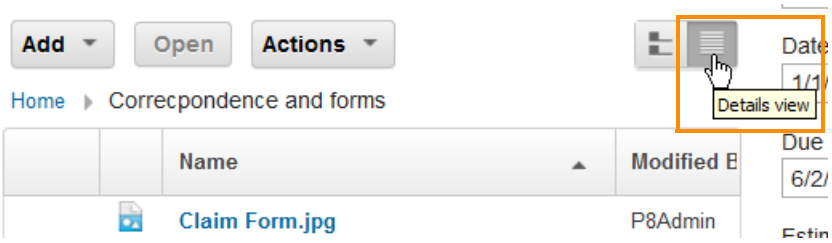
5. Check the default icon that is associated with the text document MIME type.
   a. Observe the different icons for each MIME type.
   b. Check the icon for the Notes.txt document.



6. You are going to replace this icon with a custom image.
   a. Click Close.
7. Logout of the Case client and close the browser.

Notes - Magazine View and Details View

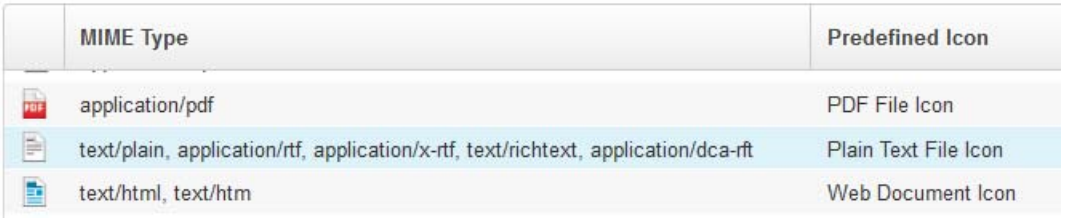
The document list in the screen capture in step 5 shows the Details View. Depending on the view that you have in your student system, the document list display might be different.



Procedure 2: Customize the icon for text MIME type

1. Start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`

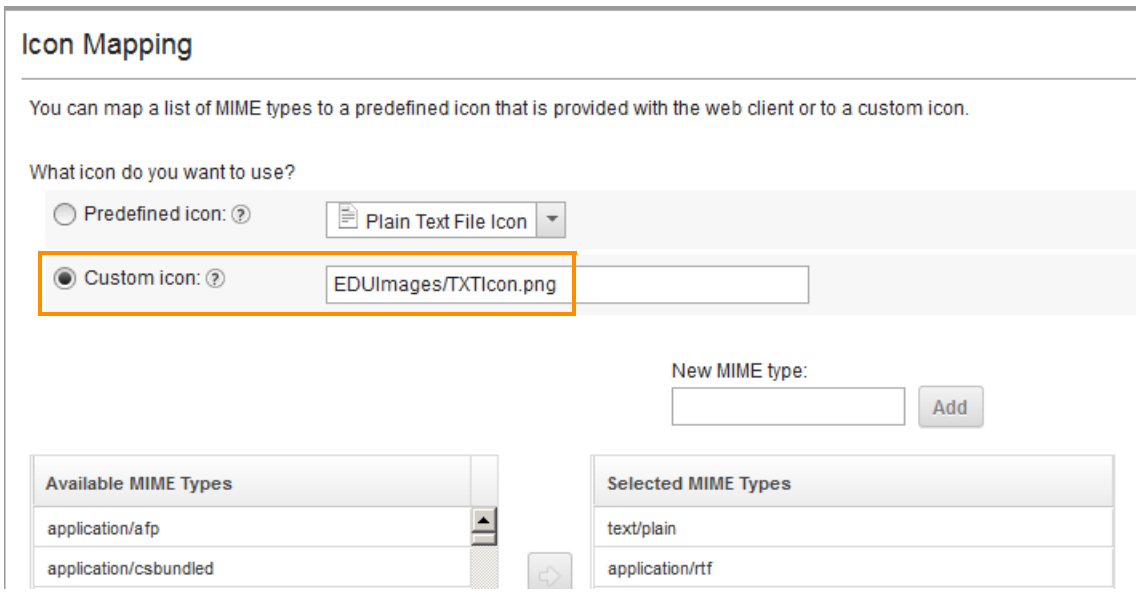
2. On the administration client, click the “Open Administration View” icon on the left pane. If this icon is not visible, refresh the browser.
3. In the Administration page, click `Settings` on the left pane.
4. In the Settings tab > Icon Mapping subtab, select `text/plain, application/rtf...` MIME type.



Hint

Observe that the list has a similar item with the MIME type “`text/html`” for Web Document Icon. For this lab, select the “Plain Text File Icon” item with the MIME type “`text/plain`”.

5. Click Edit.
   a. In the Icon Mapping window, select the “Custom icon” option.
   b. Enter `EDUImages/TXTIcon.png` as the value. This image file is already copied to your student system.

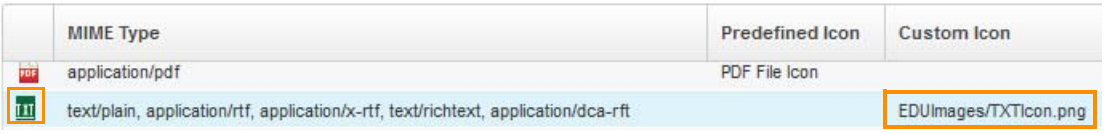


c. Click OK.

Note

You can include more MIME types to associate with this icon by moving items from the Available MIME Types list to the Selected MIME Types list. To add an item that is not in the Available list, you can enter the name in the “New MIME type” field and click Add.

6. Back in the Icon Mapping tab, verify that the text/Plain MIME type shows a new icon that you edited.
   a. The Custom Icon column shows the relative URL (path) for the image file.



7. Click Save and Close.
8. Log out of the administration tool and close the browser.

Procedure 3: Test the new custom icon

1. Repeat Procedure 1: Check the default icon, page. 1-19 to test the custom icon that is associated with the text document MIME type.
   a. Verify the new icon for the Notes.txt document.
   b. For comparison, the default icon before the customization is also shown.

2. Log out of the client and close the browser.

Troubleshooting

If the new icon is not shown, log out, clear the browser cache, close the browser and log in back.

Exercise 1.2.2: Customize Labels for IBM Case Manager Client

Introduction

In this lab, you edit an Application Label name and customize it.

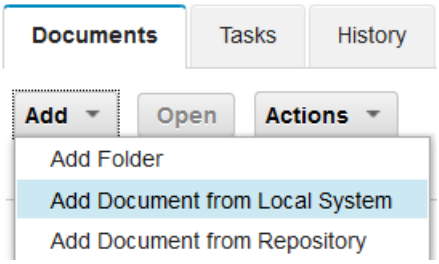
Procedures

Procedure 1: Check the default Application Label

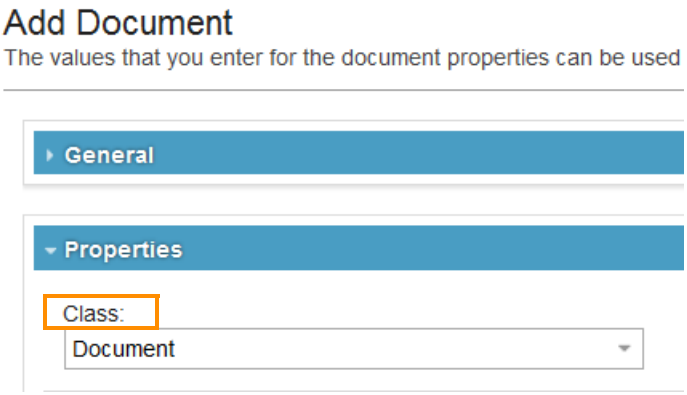
1. Start the Custom Case Manager Client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=custom`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Open a case.
   a. Click the Cases tab.
   b. In the Search section, select Policy Family Name from the list.
   c. Enter `%` in the second text box of the Search widget and click the Search button.

   d. Available cases are listed in the Case List widget.
3. Open the Case details for a case.
   a. Click the link for the case with `Smith` as the Policy Family Name value from Title column. You can select any case from the list.
   b. The Case details are shown in a tab.

4. Add a document to the case.
   a. In Documents subtab, click Add > Add Document from Local System.



5. Check the label for `Class`.
   a. In the Add Document wizard > Properties section, check that the label name for the document class is `Class`.

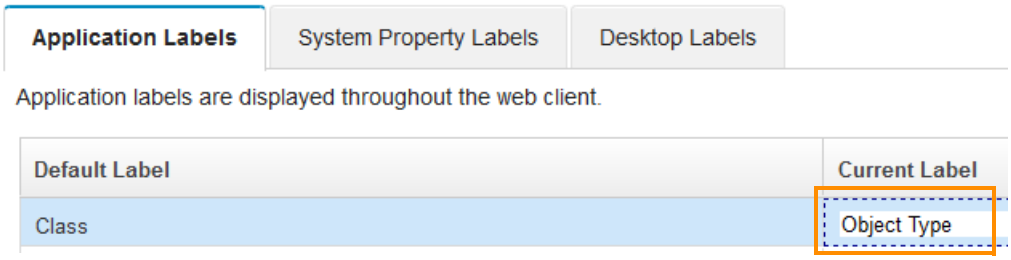


   b. You are going to change this label name to `Object Type`.
   c. Click Cancel to close the wizard.
   d. Logout of the Case client and close the browser.

Procedure 2: Customize Application Labels

1. Start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. On the administration client, click the “Open Administration View” icon on the left pane. If this icon is not visible, refresh the browser.
3. In the admin tool, click `Labels` on the left pane.
4. In the Labels tab on the right > Application Labels subtab, select `Class` in the Default Label column.

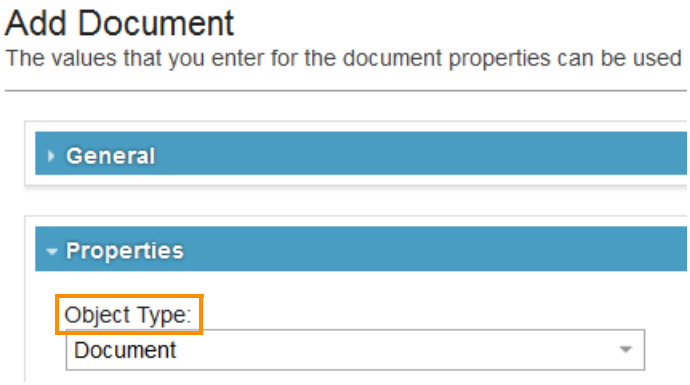
5. Double-click in the space next to that label in the Current Label column.
   a. Enter `Object Type` as the new label.



   b. Click Save and Close.
6. Log out of admin tool and close the Browser.

Procedure 3: Test the new label

1. Repeat Procedure 1: Check the default Application Label, page. 1-23 to test the new label.
2. Check the label for `Class`.
   a. In the Add Document wizard > Properties section, check that the label name for the document class is changed to `Object Type`.



   b. Click Cancel to close the wizard.
   c. Logout of the Case client and close the browser.

Troubleshooting

If the new label is not shown, log out, clear the browser cache, close the browser and log in back.

LESSON 1.3: Specify Viewers for File Types

What this lesson is about?

This lesson describes how to create a Viewer Map and associate it with the Case Manager custom client. Viewer Maps specify which viewer to use to open a file type.

What you should be able to do?

After completing this lesson, you should be able to:

- Create a Viewer Map and associate it with the Case Manager custom client.

How you will check your progress?

- Hands on labs.

Viewer Maps

Viewer maps specify which viewer is used to open each file type.

- The Case Manager administration tool includes a default Viewer Map.
  - It includes Viewer Mappings for each repository type.
  - Cannot edit the default Viewer Map.
  - Can create copies of the default Viewer Map and customize.
- You can create a custom Viewer Map for the following tasks.
  - Use a different viewer for a file type.
  - Add more file types to the mapping.
- You can create one or more Viewer Maps.
  - Can associate only one Viewer Map with each custom client.

Fallback Viewers

- Each viewer supports a specific set of file types.
  - Some viewers are not supported on every server or client platform that IBM Content Navigator supports.
- IBM Content Navigator can automatically select a different viewer to use if another viewer is included in your Viewer Map.
- This behavior is called the Fallback behavior, and the second viewer is called a Fallback viewer.
- You can include multiple Fallback Viewers in your Viewer Map.

Custom Viewers

- You can create custom viewers programmatically as a plug-in and associate it with IBM Content Navigator.

Viewers that support all MIME types

- Some viewers support only specific MIME types (Adobe Reader supports PDF type).
- The following viewers in IBM Content Navigator support all MIME types:
  - AJAX Viewer
  - Applet Viewer
  - HTML Conversion
  - PDF Conversion
  - Web Browser

Demonstrations

Create a Viewer Map

Click here to watch the demonstration.

Exercise 1.3.1: Create a Viewer Map for PDF files

Introduction

In this lab, you create a custom Viewer Map and associate it with your custom case client. In your custom Viewer Map, you replace the default viewers for the PDF files with Adobe Reader.

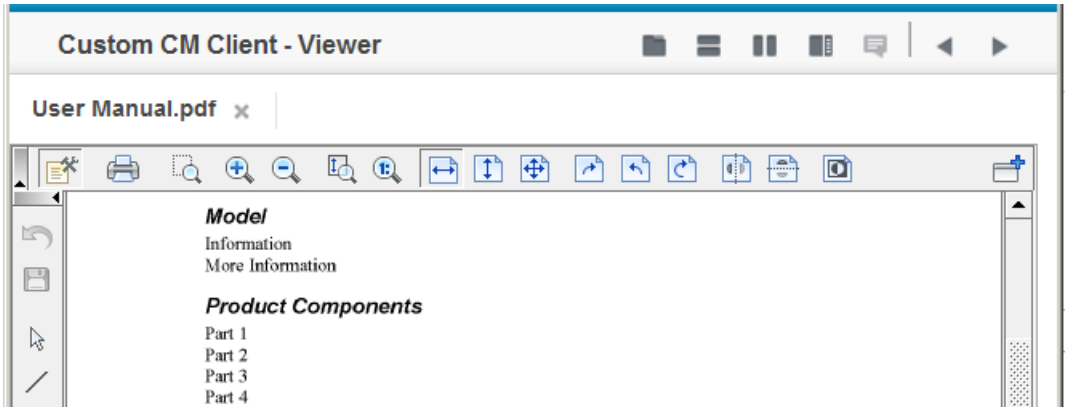
Procedures

Procedure 1: Check the existing viewer for Adobe PDF files

1. Start the custom case client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=custom`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Search for cases.
   a. Click the Cases tab.
   b. In the Search section, select Policy Family Name from the list.
   c. Enter `%` in the second text box of the Search widget and click the Search button.

   d. Available cases are listed in the Case List widget.
3. Open the documents list for a case.
   a. Click the link for the case with `Smith` as the Policy Family Name value from Title column.
   b. The details for that case are shown in a separate tab.
   c. In Documents subtab, click Correspondence and forms folder.
   d. Optionally, to see the complete list of documents, collapse the Timeline Visualizer widget.

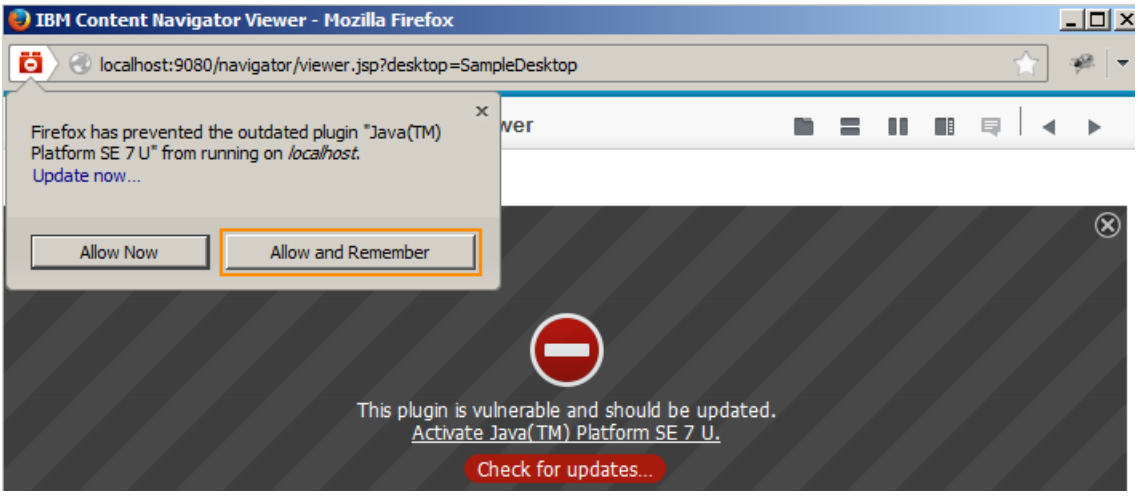
4. Check the viewer that is available to open a PDF document.
   a. Scroll down and double-click a PDF document to open it in the default viewer.
   b. Click Run if you are prompted.
   c. The Document opens in the default viewer.



5. You are going to change the default viewer that is assigned to the PDF documents.
   a. Close the Viewer.

Troubleshooting

If you get the following error when you try to open a document in the viewer, do Step 6. Otherwise, skip Procedure 2.



6. Click the red icon at the upper left corner.
   a. Click “Allow and Remember” in the window.
   b. Do not update to new Java version.

c. Close the error window by clicking X mark at the upper right corner.

d. Log out and close the browser.

Procedure 2: Create a Viewer Map

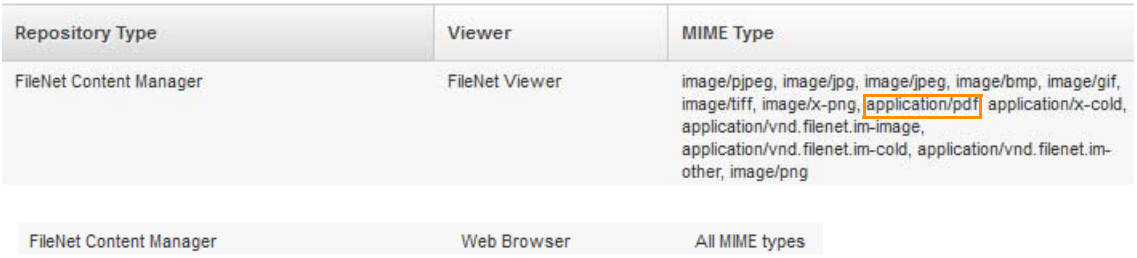
1. Start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. On the administration client, click the “Open Administration View” icon on the left pane. If this icon is not visible, refresh the browser.
3. Click `Viewer Maps` on the left pane.
4. Create a Viewer Map.
   a. In the Viewer Maps tab on the right pane, click New Viewer Map.

      The New Viewer Map contains a copy of all the mappings from the default Viewer Map.

   b. In the New Viewer Map tab, enter `EDU Viewer Map` in the Name field.

      The ID value is populated automatically.

   c. Edit the description. (Example: `It maps Adobe Reader for the PDF files`).
5. Check the mappings for the FileNet Content Manager repository.
   a. There are two mappings in the list. Observe the application/pdf MIME type



Repository Type	Viewer	MIME Type
FileNet Content Manager	FileNet Viewer	image/pjpeg, image/jpg, image/jpeg, image/bmp, image/gif, image/tiff, image/x-png, application/pdf, application/x-cold, application/vnd.filenet.im-image, application/vnd.filenet.im-cold, application/vnd.filenet.im-other, image/png
FileNet Content Manager	Web Browser	All MIME types

6. Remove the application/pdf MIME type from the FileNet Viewer mapping.
   a. Select the FileNet Viewer row for FileNet Content Manager and click Edit.
   b. In the Mapping page > Selected MIME Types pane, scroll down and select the application/pdf.
   c. Move it from the Selected MIME Types pane to the Available MIME Types pane by clicking the arrow. Click OK.

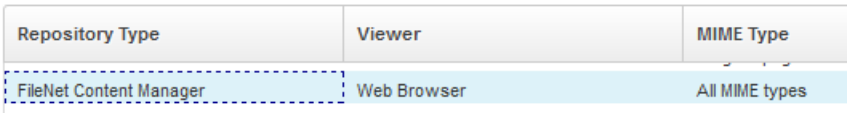
      This step removes the option to view the PDF files in the FileNet Viewer.

   d. In the New Viewer Map, verify that the change is updated. The application/pdf MIME type is removed from the list.

Note

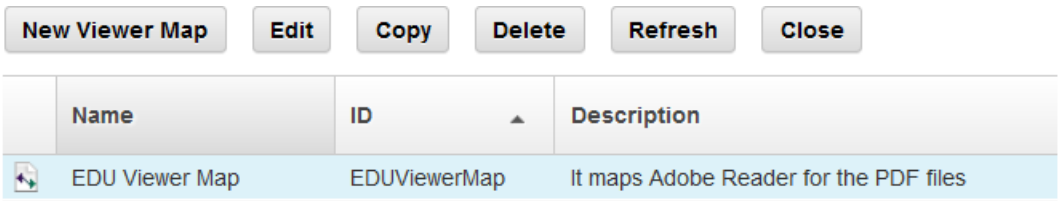
The Case client selects Web Browser as the first Fallback Viewer. For the case client to use the new mapping, you must remove the application/pdf MIME type from the Web Browser mapping.

7. Remove the application/pdf MIME type from the Web Browser mapping for the FileNet Content Manager.
   a. Select the Web Browser Viewer row for the FileNet Content Manager and click Edit.



   b. In the Mapping page, clear the “All MIME types” option.
8. Move all the items in the Available MIME Types pane except application/pdf.
   a. Select all the items in the Available MIME Types pane by pressing shift and click.
   b. Move them to the Selected MIME Types pane by clicking the forward arrow.
   c. In the Selected MIME Types pane, select the `application/pdf`.
   d. Move it from the Selected MIME Types pane back to the Available MIME Types pane by using the arrow.
   e. Click OK.

      This step removes the option to view the PDF files in the Web Viewer.
   f. In the New Viewer Map, verify that the change is updated. A list of MIME types is shown.
9. Create a Mapping for pdf.
   a. Click New Mapping.
   b. In the New Mapping page, select FileNet Content Manager for the “Repository type” field.
   c. Select Adobe Reader for the Viewer field.
   d. Move `application/pdf` from the Available MIME Types pane to the Selected MIME Types pane by using the arrow. Click OK.
   e. In the New Viewer Map tab, click Save and Close.
10. In the Viewer Maps, verify that the newly created map is listed.



11. Leave the application open for the next procedure.

Procedure 3: Assign the new Viewer Map to your case client

You replace the default Viewer Map that is assigned to your case client with the new one that you created.

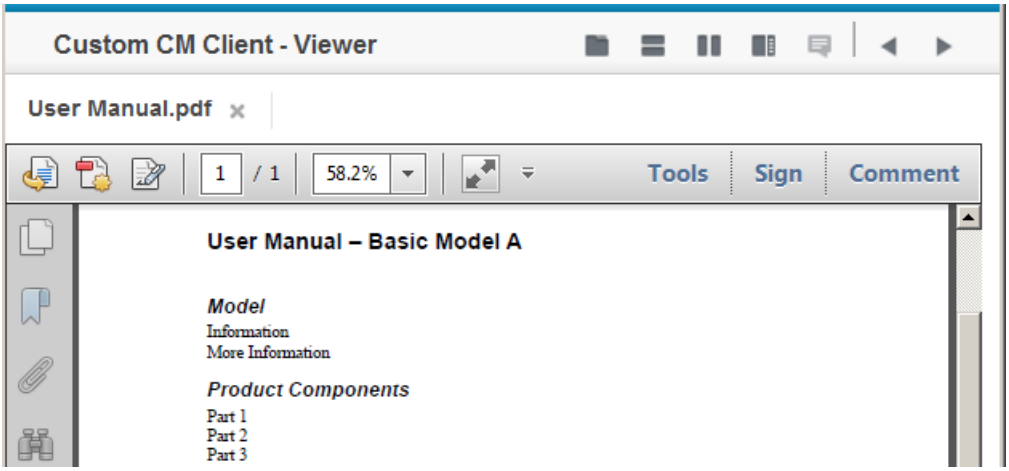
1. Click the Desktops tab. Select Custom CM Client and click Edit.
2. In the General tab > Desktop configuration section, select the new Viewer Map from the list.

3. Click Save and Close.
4. Log out of the Case client and close the browser.

Procedure 4: Test the new Viewer Map

In this procedure, you test the new Viewer Map by opening a PDF file.

1. Repeat Procedure 1: Check the existing viewer for Adobe PDF files, page. 1-29, to open a PDF document.
   a. Verify that the PDF file is opened in the Adobe Reader that is associated with Firefox.



2. Log out of the case client and close the browser.

Hint

If the Adobe Reader tools are not visible, click the document and select “Show Adobe Reader toolbar”.

Exercise 1.3.2: View the Microsoft Word documents in the FileNet Viewe

Introduction

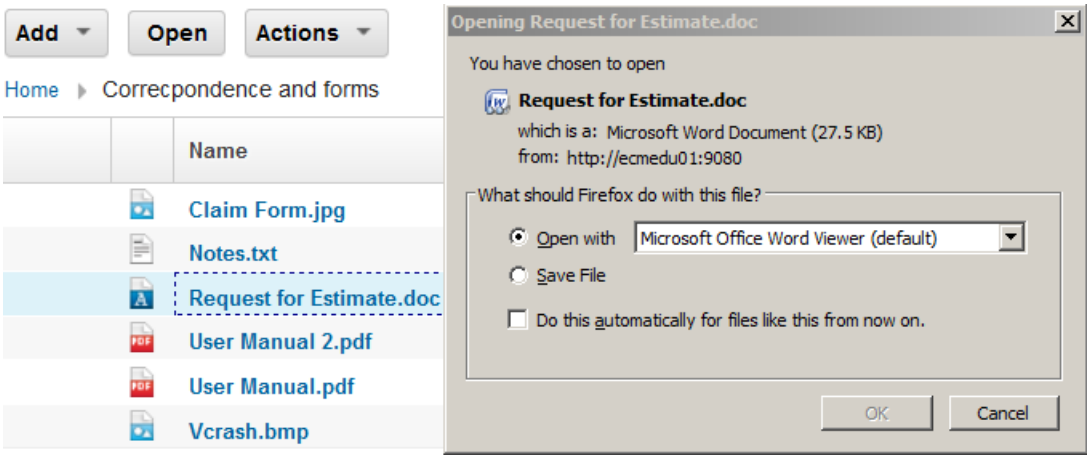
The default setting for the Word documents is to open them in Microsoft Word or Word Viewer from a web browser. Some of your Business Users must be able to open the Microsoft Word documents in the default FileNet Viewer for a quick reviewing and adding annotations.

In this lab, you modify the custom Viewer Map that you created in the previous exercise to view the Microsoft Word documents in the FileNet Viewer.

Procedures

Procedure 1: Check the existing viewer for Microsoft Word documents

1. Check the default viewer that is available to open a Microsoft Word document for a case.
   a. Repeat Procedure 1: Check the existing viewer for Adobe PDF files, page. 1-29 from the previous exercise to open a Word document.
   b. Verify that the default Viewer for the Word document is Microsoft Office Word Viewer.



2. Click Cancel to close the dialog.
   a. You are going to change the default viewer that is assigned to the Word documents.
   b. Log out and close the browser.

Procedure 2: Edit the custom Viewer Map

1. Start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. On the administration client, click the “Open Administration View” icon on the left pane. If this icon is not visible, refresh the browser.
3. Click `Viewer Maps` on the left pane.
4. Edit the custom Viewer Map that you created earlier.
   a. In the Viewer Maps tab on the right pane, select EDU Viewer Map and click Edit.
   b. In the EDU Viewer Map tab, edit the description. (Example: `Viewer map for the PDF and Word files`).
5. Check the mappings for the Case Manager (FileNet Content Manager repository).
   a. Observe the `application/msword` MIME type for the two mappings in the list.
   b. Web Browser (it has the `application/msword` MIME type)
   c. FileNet Viewer (it does not have the `application/msword` MIME type)
6. Add the `application/msword` MIME type to the FileNet Viewer mapping.
   a. Select the FileNet Viewer row (FileNet Content Manager) and click Edit.
   b. In the Mapping page > Available MIME Types pane, select the application/msword.
   c. Move it from the Available MIME Types pane to the Selected MIME Types pane by using the arrow.
   d. Click OK.

      This step adds the option to view the Microsoft Word documents in the FileNet Viewer.
   e. In the EDU Viewer Map, verify that the `application/msword` MIME type is added.
7. Remove the `application/msword` MIME type from the Web Browser mapping for the FileNet Content Manager.

Note

The IBM Content Navigator (Case client) selects Web Browser as the first Fallback Viewer. For the Case client to use the new mapping, you must remove the application/msword MIME type from the Web Browser mapping.

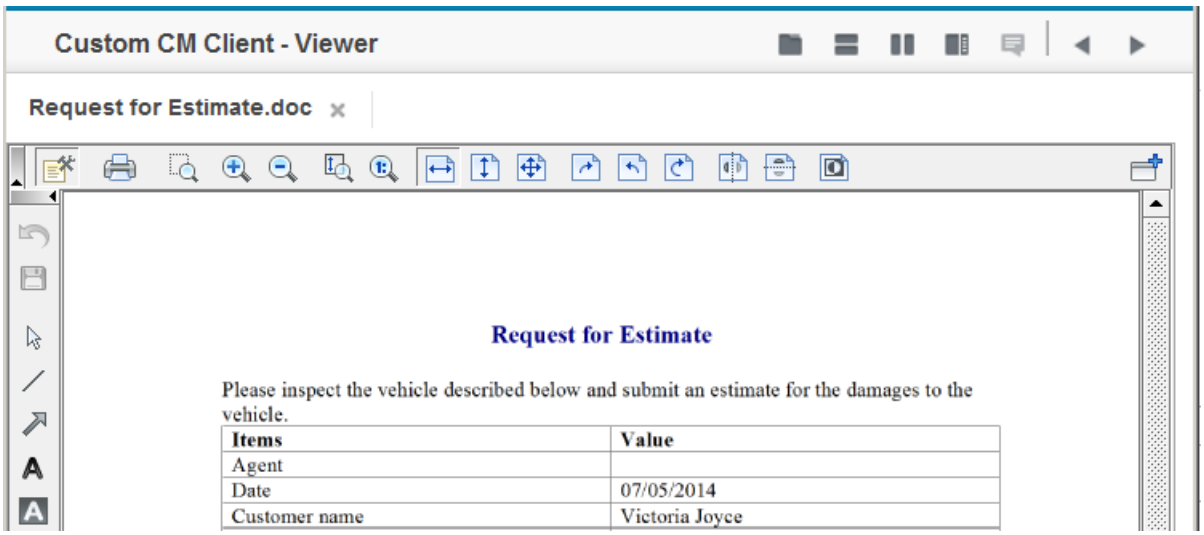
   a. Select the Web Browser Viewer row for the FileNet Content Manager click Edit.
   b. In the Mapping page > Selected MIME Types pane, select the application/msword.
   c. Move it from the Selected MIME Types pane to the Available MIME Types pane by using the arrow.
   d. Click OK. This step removes the option to view the Microsoft Word documents in the Web Viewer.

e. In the EDU Viewer Map, verify that the `application/msword` MIME type is removed.

8. Click Save and Close.
9. Log out of the application and close the browser.

Procedure 3: Test the modified custom Viewer Map

1. Repeat Procedure 1: Check the existing viewer for Adobe PDF files, page. 1-29 from the previous exercise to open a Word document.
2. Verify that the Word document is now opened in FileNet Viewer as you configured.



3. Close the Viewer.
4. Log out of Case client and close the browser.

LESSON 1.4: Create and Customize Help Topics

What this lesson is about?

This lesson describes how to edit or add custom help topics to the IBM Case Manager help system and how to create a help plug-in.

What you should be able to do?

After completing this lesson, you should be able to:

- Edit the existing help topics.
- Create a help plug-in.
- Add custom help topics to the IBM Case Manager help system.

How you will check your progress?

- Hands on labs.

References

IBM Case Manager Version 5.2 Information Center

IBM Redbooks publication: Advanced Case Management with IBM Case Manager

Add custom help topics

Help system for IBM Case Manager

You can modify the online help files or add your own help files for your users in IBM Case Manager.

- The help system for IBM Case Manager uses the formatting of Eclipse plug-ins to display content.
  - The help files are created in Darwin Information Typing Architecture (DITA) XML.
  - They are converted to XHTML and use a cascading style sheet named swg_info_common.css.
- Help files are installed in the following location: IBM Case Manager_installation_directory /help/content.
  - They are then copied to the network shared directory when you configure IBM Case Manager.
  - You must modify the files in the network shared directory for changes to be reflected in the help system.
- The location for the network shared directory is specified on the “Set properties for development environment” page in the Create Profile wizard.
  - This wizard is run from the IBM Case Manager administration client.
  - The default value for network shared directory on a Windows server: C:\Program Files(x86)\IBM\CaseManagement\configure\properties\help\content\.

Create new help topics

You can create new help files to add to the IBM Case Manager help system, if you want to add custom instructions in a separate file.

If you introduce a new cascading style sheet (CSS) for the help files, modify the HTML to use the new CSS file.

Note

Any new files are overwritten in this location (network shared directory that you configured for IBM Case Manager) if you upgrade to a new version of IBM Case Manager.

Creating a help plug-in

Help plug-in

You can create a new help plug-in for the IBM Case Manager help system to add a section of custom content for your users.

- If you have specific instructions for your users, you might want to add a section to the help system by creating a new help plug-in.
- To create your own help plug-in:
  1. Create a subdirectory for your new or modified help in the following directory: IBM Case Manager_installation_directory/network_shared_directory/help/conten

Important

Placing your topics in a separate subdirectory ensures that when an upgrade or fix pack is applied, your help are not overwritten.

  2. Create all content files in the Extensible Hypertext Markup Language (XHTML) format.
  3. Create table of contents files that describe the navigation for your topics. The following example shows a simple table of contents in an XML file called toc.xml.

```
<?xml version="1.0" encoding="UTF-8">
  <toc label="My Custom Help">
    <topic label="Help Topic 1" href="Help1.htm"></topic>
    <topic label="Help Topic 2" href="Help2.htm"></topic>
    <topic label="Help Topic 3" href="Help3.htm"></topic>
  </toc>
```

  4. Create the plugin.xml file that extends the org.eclipse.help.toc extension point and specifies table of contents files.

     Every plug-in requires a file that is called plugin.xml to identify the plug-in contents to the system. The id for the <plugin> element should match the subdirectory name.

```
<?xml version="1.0" encoding="utf-8"?>
<plugin id="com.mycustomhelp.doc" name="My Custom Help"
provider-name="CompanyName" version="1.0">
  <extension point="org.eclipse.help.toc">
    <toc file="toc.xml" primary="true"/>
  </extension>
</plugin>
```

  5. Restart the application server.

Demonstrations

Create a help plug-in

Click here to watch the demonstration.

Exercise 1.4.1: Edit the existing help topics

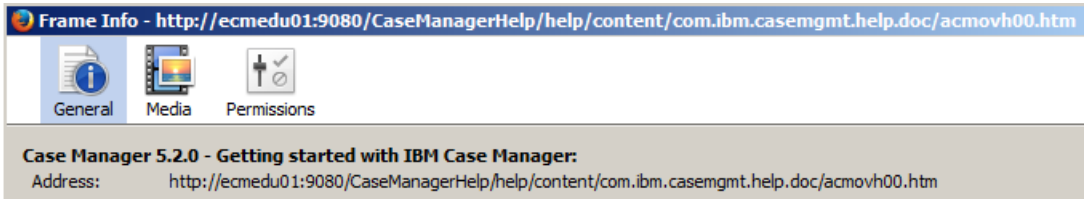
Introduction

In this lab, you edit the existing help topics in the IBM Case Manager help system.

Procedures

Procedure 1: Edit the existing help topics

1. View the default help system for IBM Case Manager.
   a. In a Firefox browser, go to the URL:
   `http://ecmedu01:9080/CaseManagerHelp/index.jsp`
   b. Expand the IBM Case Manager help node in the left pane and click the “Getting Started with IBM Case Manager” node.
2. Select a page in help system that you want to edit.
   a. For this exercise, you use “Getting Started with IBM Case Manager” page.
   b. Get the URL for this Frame by right-clicking on the page and select This Frame > View Frame Info.



   c. From the Address field, note down the package and file name to edit:
   `/com.ibm.casemgmt.help.doc/acmovh00.htm`
   d. Close the Frame Info page and leave the help system open in the browser to test it after editing.
3. Go to the following directory and open the `acmovh00.htm` file in a text editor (Notepad+
   `C:\Program Files (x86)\IBM\CaseManagement\configure\properties\help\content\com.ibm.casemgmt.help.doc\`
4. Edit the file to add a section for a custom topic.
   a. For this section, you can add any changes that you want or do the Steps 4b.
   b. Enter the text on Line 44 and 45 as shown in the following screen capture.

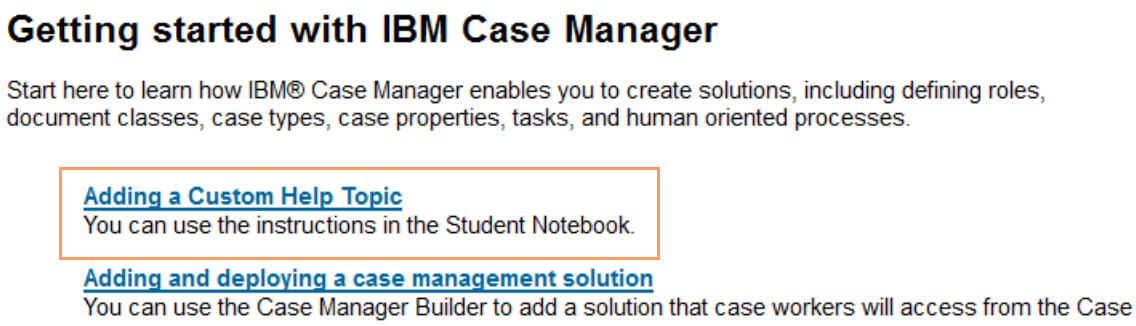
```
44  <li class="link ulchildlink"><span class="ulchildlinktext"><a href="acmsdh00.htm">
    Adding a Custom Help Topic </a></span><br />
45  You can use the <span class="keyword">instructions</span> in the Student Notebook.
46  <li class="link ulchildlink"><span class="ulchildlinktext"><a href="acmsdh00.htm">
    Adding and deploying a case management solution</a></span><br />
47  You can use the <span class="keyword">Case Manager Builder</span> to
```

Hint

Optionally, copy the text from `C:\ICM\CustomHelpTopics\acmovh00_Solution.htm` and paste it.

c. Save your changes and close the file.

5. Test the “Getting Started with IBM Case Manager” page.
   a. Repeat Step 1 to access the help page and refresh the browser.
6. Verify that the edits that you added is reflected on the page and it looks like the following screen capture.



7. Close the browser.

Exercise 1.4.2: Create a help plug-in

Introduction

IBM Case Manager installation provides a default help plug-in for the Case Manager Client in the installation directory. They are not yet copied to the network shared directory and are not available online.

In this lab, you use the default plug-in as a base for your plug-in. You explore, edit, and deploy the plug-in.

Procedures

Procedure 1: Creating a help plug-in

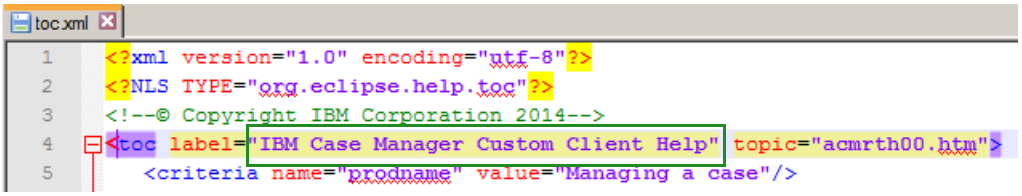
1. Stop the Web Application Server
   a. Double-click the `Stop the Server1.bat` file in the WebSphere Admin folder on the desktop.
   b. Wait for the Stop the server page to close.
2. To create a subdirectory for your new or modified h
   a. Copy the `com.ibm.casemgmt.client.doc` directory from the following location:
      `C:\Program Files (x86)\IBM\CaseManagement\help\content`
   b. Paste it to the following location:
      `C:\Program Files (x86)\IBM\CaseManagement\configure\properties\help\content`
   c. Rename the package to `com.ibm.casemgmt.client.custom.doc`
3. Sample content files are already created in the Extensible Hypertext Markup Language (XHTML) format.
4. Edit the table of contents file:

Note

You must have a file for table of contents to show as a navigation tree in the left pane. A table of contents file (acmrt.xml) that describe the navigation for the topics is provided.

   a. Rename the `acmrt.xml` file to `toc.xml` in the following directory.
      `C:\Program Files (x86)\IBM\CaseManagement\configure\properties\help\content\com.ibm.casemgmt.client.custom.doc`
   b. Open the `toc.xml` file and view the content.

c. Change the top-level label “Managing cases” in line 4 to “`IBM Case Manager Custom Client Help`”.



d. Save your changes and close the file.

5. Edit the plugin.xml file (from the same directory as the previous step) as shown in the following text to specify your subdirectory and `toc` file. Required changes are shown in bold font.

Note

Every help plug-in requires a file that is called `plugin.xml` to identify the plug-in contents to the system. The id for the <plugin> element should match the subdirectory name. This plugin.xml file must extend the org.eclipse.help.toc extension point and specify the table of contents files.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml eclipse version="3.0"?>
<plugin id="com.ibm.casemgmt.client.custom.doc" name="My Custom Help"
provider-name="IBM" version="5.2">
   <extension point="org.eclipse.help.toc">
      <toc file="toc.xml" primary="true"/>
   </extension>
   ...
```

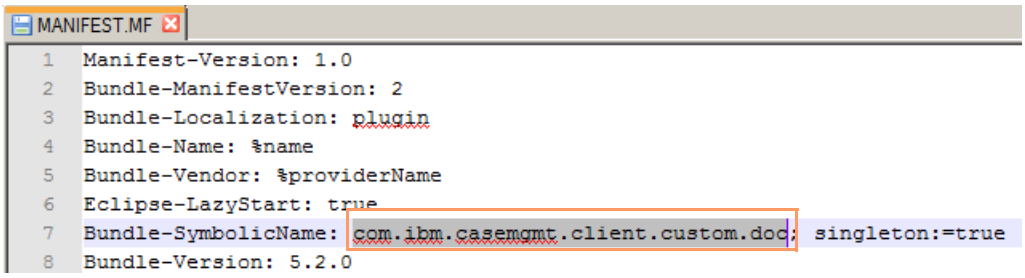
Hint

Optionally, copy the text from `C:\ICM\CustomHelpTopics\plugin_Solution.xml` and paste it.

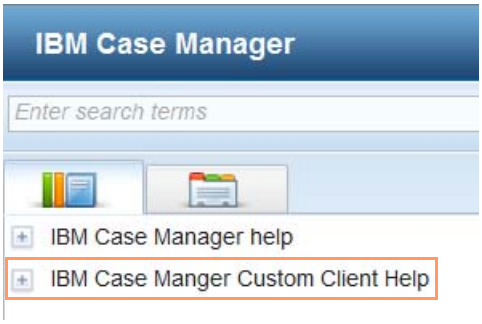
e. Save your changes and close the file.

6. Edit the `MANIFEST.MF` file in the following directory to provide the custom bundle name:

   `C:\Program Files (x86)\IBM\CaseManagement\configure\properties\help\content\com.ibm.casemgmt.client.custom.doc\META-INF`



7. Start the Web Application Server:
   a. Double-click the `Start Server1.bat` file in the WebSphere Admin folder on the desktop.
   b. Wait for the Start the server page to close.
8. Verify that the help plug-in is deployed and the new help topic is shown in the navigation.
   a. In a Firefox browser, go to the URL: `http://ecmedu01:9080/CaseManagerHelp/index.jsp`
   a. Click the `IBM Case Manager Custom Client Help` node. Check that the help topic is shown in the right pane.
   b. For comparison, the help system before the customization is also shown.



9. Close the browser.

Procedure 2: Add more topics to the custom help system

In this procedure, you add more topics to your custom help system that you deployed in the previous procedure.

1. Create a custom help topic (in XHTML format as `htm` file) in a text editor and save it in the following directory: `C:\Program Files`

(x86)\IBM\CaseManagement\configure\properties\help\content\com.ibm.casemgmt.client.custom.doc

a. For this exercise, name the file as MyCustomHelpTopic.htm

b. You reference this file name in the following step.

Hint

Optionally, copy the MyCustomHelpTopic.htm file from C:\ICM\CustomHelpTopics folder to the directory in Step 1.

2. Declare the new help file in the toc.xml file.

   a. Open the toc.xml file in the following directory:

   C:\Program Files (x86)\IBM\CaseManagement\configure\properties\help\content\com.ibm.casemgmt.client.custom.doc

   b. Add a line with your custom help file name (towards the end of the file) as shown in the following screen capture. You can provide any text for the label field.

```
15      </topic>
16      <topic label="Search tips for cases" href="acmrth04.htm"/>
17      <topic label="Search tips for documents" href="acmrth05.htm"/>
18      <topic label="Filing a Claim" href="MyCustomHelpTopic.htm"/>
19  </topic>
20  </toc>
```

   c. Save your changes and close the file.

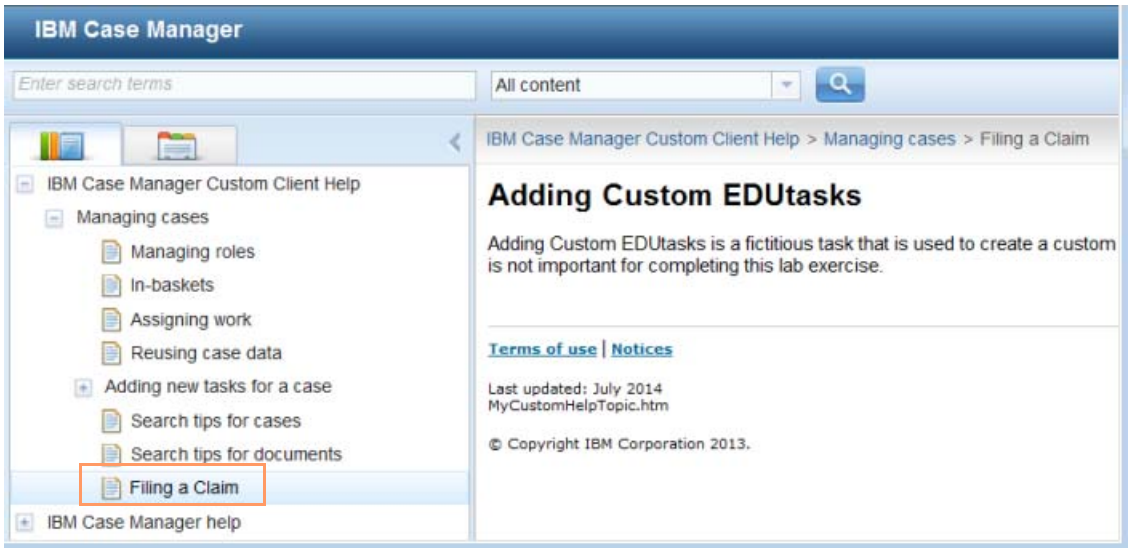
3. Verify that the new topic is added to the custom help system.

   a. In a Firefox browser, go to the URL:
   http://ecmedu01:9080/CaseManagerHelp/index.jsp

   a. Expand the IBM Case Manager Custom Client Help > Managing Cases node.

   b. Check that the “Filing a Claim” help topic is shown in the left pane navigation.

   c. Click that link and the custom page that you added is shown in the right pane.



Troubleshooting

If the new topic does not show up in the help system, restart the Web Application Server.

LESSON 1.5: Customize Toolbar and Menu

What this lesson is about?

This lesson describes how to implement actions by adding buttons to the toolbar, and add custom actions to the Menu.

What you should be able to do?

After completing this lesson, you should be able to:

- Customize the toolbar to implement actions.
- Add a custom action as a menu item.

How you will check your progress?

- Hands on labs.

References

IBM Case Manager Version 5.2 Information Center

IBM Redbooks publication: Advanced Case Management with IBM Case Manager

Toolbar widgets

Case Toolbar widget

You use the Case Toolbar widget to specify the actions that case workers can take for cases.

- These actions can be implemented as buttons or menu buttons in the toolbar.
- The buttons and menu buttons that are included by default in the Case Toolbar widget depend on the type of page that has the widget.
- You can edit the settings for the Case Toolbar widget to add, remove, and edit the buttons and menu buttons that are available to case workers.

Toolbar widget events

- The Case Toolbar widget handles events to display and process actions that the user can perform for a case.
  - Example: The Case Toolbar widget handles an incoming event to display the properties for a case type so that the user can create a case.
- The Case Toolbar widget provides incoming events to handle the data that is received from other widgets.
- The Case Toolbar widget publishes events if certain actions are configured for the widget.
- Example: If the Add Case action is configured for the Toolbar widget, the widget publishes the Open new case page event for that action.

IBM Case Manager pages with toolbar widget

The following table shows example of the Case Manager pages that includes the Toolbar widget by default. The table also shows the available toolbar buttons for some of the default pages.

Page Name	Toolbar buttons
Cases page	Add Case
Work page	- Add Case - Manage Roles
Add Case page	- Save Case and Close Page - Close Case Page

Case Details page	- Add Comment to Case - Add Task - Close Case Page - Save Case - Split Case
Split Case page	- Save Case and Close Page - Close Case Page
Work Details page	- Add Comment to Work Item - Close Work Details Page - Dispatch Work Item - Open Next Work Item - Reassign Item - Save Work Item

Demonstrations

Customize the toolbar to implement actions

Click here to watch the demonstration.

Add a custom action as a menu item

Click here to watch the demonstration.

Exercise 1.5.1: Customize the toolbar to implement actions

Introduction

In this exercise, you customize the toolbar to implement actions as buttons. You add a button to show the link to a case, and another button to open a web page.

Procedures

Procedure 1: Create a custom page

In this procedure, you copy the default Cases page and modify it. A solution for this lab is already created.

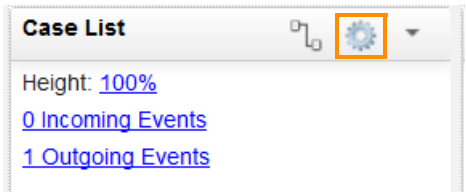
1. In Firefox, log on to IBM Case Manager Builder as an administrative user.
   - URL: `http://ecmedu01:9080/CaseBuilder`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.

   Hover the mouse over the solution to see the links.
3. Create a custom page:
   a. Open the Pages tab and expand the Solution Pages.
   b. Click and hover the mouse over the Cases page name.
   c. Select the copy icon on the right side of the page.
   d. In the resulting page, edit the name to `Custom Toolbar Menu` for your new page and click OK to create the copy.
   e. Save your work by clicking Save at the top of the page.
   f. Leave the page open for the next procedure.

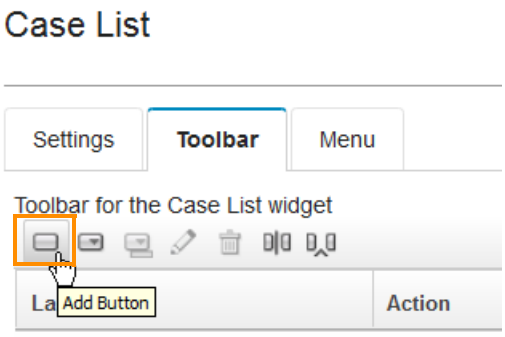
Procedure 2: Edit the page to customize the toolbar

In this procedure, you implement actions as buttons in the toolbar.

1. In the Pages tab, click the Custom Toolbar Menu page to edit it in Page Designer.
2. Click the “Edit Settings” icon for the Case List widget in the middle pane.



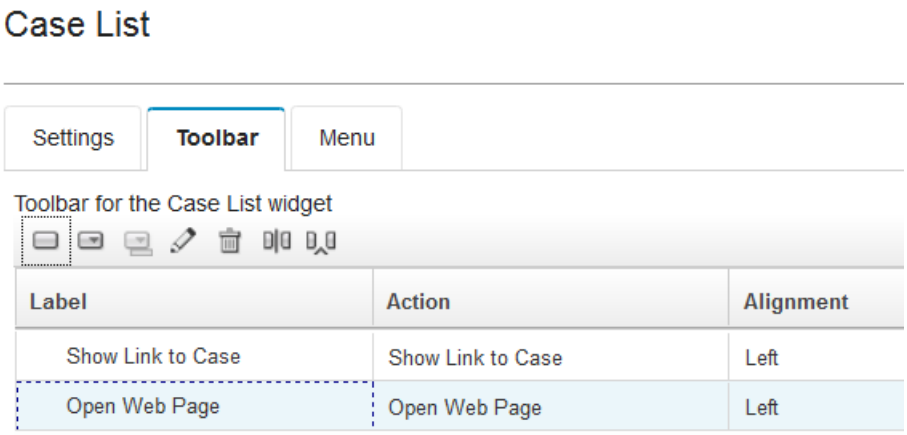
   a. Click the Toolbar tab.
3. Click the Add Button.



4. Select “Show Link to Case” from the list for the Action field.
   a. Verify that “Show Link to Case” is shown for the label.
   b. Click OK.

5. Validate that you see your newly added toolbar action.
6. Repeat step 3-5 to add another button with the following values:
   - Action: `Open Web page`
   - URL: `https://ecmedu01:9043/ibm/console/logon.jsp`

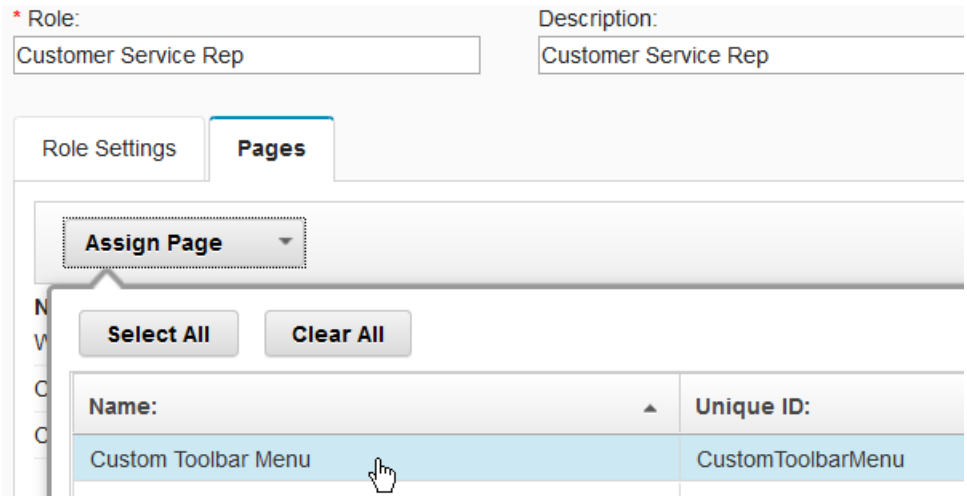
7. Verify that the list has the two actions that you added as shown in the screen capture.



8. Click OK at the end of the dialog to exit the Case List settings window.
   a. Click Save and then click Close at the upper right of the Page Designer window.
   b. Leave the solution open for the next procedure.

Procedure 3: Assign the custom page to a role

1. Your solution is already opened in the Case Builder. Open the Roles tab.
2. Click the Customer Service Rep role link.
   a. Open the Pages subtab.
   b. Click Assign Page.
   c. Select the Custom Toolbar Menu Page.



   d. Click OK to close the dialogue window.

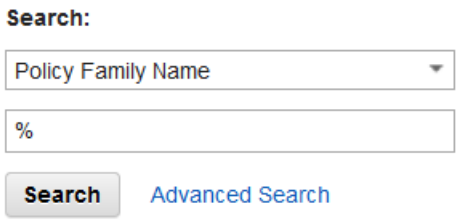
3. Verify that your page is listed in the Pages tab.
   a. Click OK All to accept the changes to the role.
   b. Click the “Save and Close” button on the solution to exit the solution editor.
   c. Leave the Case Manager Builder open for the next procedure.

Procedure 4: Redeploy the solution

1. In the Manage Solutions page, select Lab Claims Solution and hover the mouse over.
   a. Click Commit to save the changes to the repository.
   b. Click “Commit My Changes” in the Confirmation window.
2. Select the solution again, hover the mouse over, and click Deploy.
   c. Wait for the green check mark to appear next to the solution.
3. Hover the mouse over the solution and click Test to open Case Manager Client.

Procedure 5: Test the customized toolbar

1. Click the Custom Toolbar Menu tab to open the custom page.
2. Search for cases.
   a. In the Search section, select Policy Family name from the list.
   b. Enter % in the next text box of the Search widget and click the Search button.

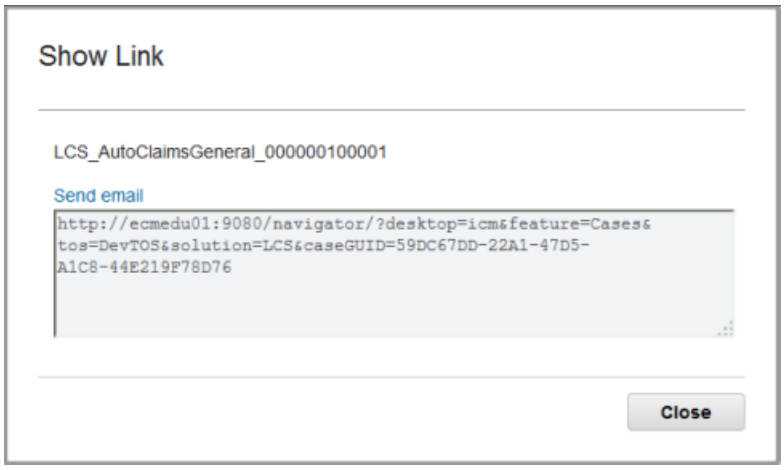


   c. Available cases are listed in the Case List widget.
3. Validate the Show Link to Case button.
   a. Select a case in the Case List widget and click the Show Link to Case button.

Hint

If you click the Title link of a case, the case opens in the Case Details page. Select a case by clicking other columns.

   b. Verify that you get a dialog page with the direct link to the case.



4. Optionally, check the link.
   a. Open a new browser tab.
   b. Copy and paste the URL into a new browser tab.
   c. Verify that the Case Details page opens for the case that you selected.
   d. Close the browser tab.
5. Validate the Open Web Page button.
   a. Click the button.
   b. Verify that the WebSphere Integrated Solutions Console opens in a new browser tab.
6. Close the browser tabs and the case client window.

Exercise 1.5.2: Add a custom action as a menu item

Introduction

In this exercise, you add a custom script action to a menu item on the default Case List widget. This script checks against the Case Title that you entered to see whether it matches with a case in the set of available cases.

Procedures

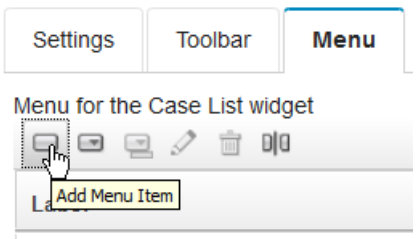
Procedure 1: Open the custom page

For this exercise, you reuse the custom page that you created in the previous exercise, and edit the page.

1. Refer to the steps in Procedure 1: Create a custom page, page. 1-53 to open the solution in Case Manager Builder.
2. Open the Pages tab.
   a. Expand the Solution Pages.
   b. Click the Custom Toolbar Menu page to edit it in Page Designer.

Procedure 2: Edit the page to add the menu item

1. In Page Designer > Custom Toolbar Menu page, click the “Edit Settings” icon for the Case List widget in the middle pane.
2. Click the Menu tab.
   a. Click the “Add Menu Item” button.



3. Select `Script Action` from the list for the Action field.
4. Optionally, edit the value for your label (Example: `Flag Special Case`).
5. Enter the following JavaScript for the execute section.

Hint

Optionally, copy the text from the C:\ICM\Menu Item Script\MenuItemScript.txt file and paste it.

```
var x = this.getActionContext("CaseReference");
var c = "LCS_AutoClaimsGeneral_000000100001";
if(c == x[0].getCaseTitle()){
   alert("This is a special case, please review this case.");
}
else{
   alert("This is a normal case, please proceed.");
}
```

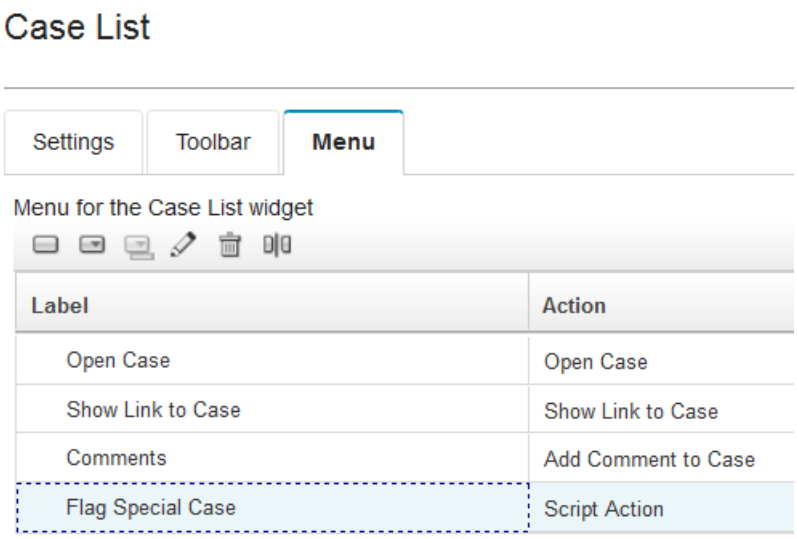
Note

A case with the case title value "`LCS_AutoClaimsGeneral_000000100001`" is added to your system and used as an example in the code. You can optionally create a case and use the Title of your case.

How does this code work?

- This script checks against the Case Title to see whether it matches a case in the variable set (A list of available cases).
- You retrieve the value of a list of cases, as an array, from the Case List widget in the `this.getActionContext("CaseReference");` line in this script.
- Then, you add an `if` statement to check against the first element in the array.

6. The OK button is not visible. Scroll down and click OK.
7. Verify that your Flag Special Case menu item is listed for the Case List widget.



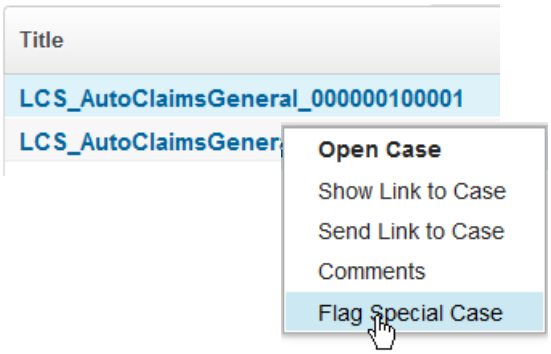
8. Click OK to exit out of the Case List dialog window.
   a. Click Save and then click Close at the upper right of the Page Designer window.
   b. Click “Save and Close” on the solution to exit the solution editor.
   c. Leave the Case Manager Builder open for the next procedure.

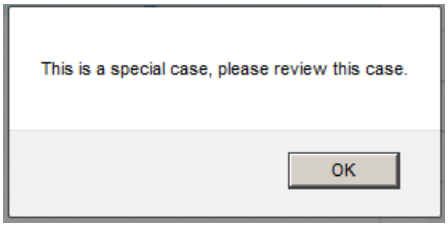
Procedure 3: Redeploy the solution

1. Refer to the steps in Procedure 4: Redeploy the solution, page. 1-56 to redeploy the solution in Case Manager Builder and to open Case Manager Client.

Procedure 4: Test the menu item

1. In Case Manager Client, click the Custom Toolbar Menu tab to open the custom page.
2. Search for cases.
   a. In the Search section, select Policy Family name from the list.
   b. Enter `%` in the next text box of the Search widget and click the Search button.
   c. Available cases are listed in the Case List widget.
3. Validate your custom script action in the menu item.
   a. Right-click the case with the Title value that you used in the code and select Flag Special Case from the menu (Policy Family Name; Smith).
   b. Verify that a dialog box is shown with the text that you added in the code: “`This is a special case, please review this case`.”





c. Click OK to close the dialog box.

4. Verify that when you select another case, a different message is shown as you specified in the code.
   a. Right-click the case with the Policy Family Name value “Beckner” and select Flag Special Case from the menu.
   b. Validate that there is a different message.
   c. Log out of the Case Manager Client and Case Manager Builder applications; Close the browser windows.

2

Use Scripts to Customize Case Manager Client

This unit provides guidance for customizing IBM Case Manager Client with the use of Script Adapter and scripts.

LESSON 2.1: IBM Case Manager JavaScript API Overview

What this lesson is about?

This lesson provides an overview of IBM Case Manager JavaScript API.

What you should be able to do?

After completing this lesson, you should be able to:

- Use IBM Case Manager JavaScript API.
- Use Script Adapter widget to customize the IBM Case Manager Case client.

How you will check your progress?

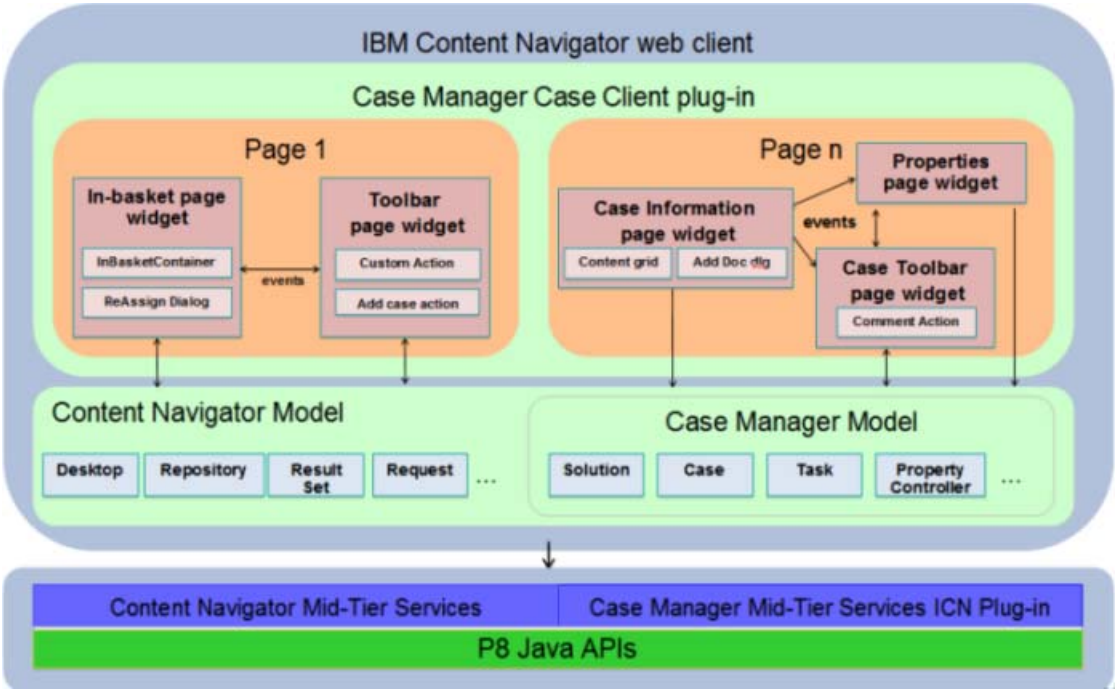
- Hands on labs.

IBM Case Manager Case Manager Development Architecture

Architectural diagram

IBM Case Manager extends the IBM Content Navigator, and provides Case Manager model classes, page widgets, and utility classes (dialogs and actions).

It uses a Model-View-Controller (MVC) architecture.



- IBM Case Manager mid-tier services are built on IBM FileNet P8 Java APIs as an IBM Content Navigator plug-in.
- IBM Case Manager model layer consists of case management objects.
  - Example: Solution, Case, Task, and Property Controller objects.
- IBM Case Manager visual layer API provides classes that represent the following features:
  - Page widgets (example: Case Search, Case List, and In-basket)
  - Dialogs (example: Comments and Reassign)
  - Actions that are configured on toolbars and menus.

IBM Case Manager API toolkits

Overview of API toolkits

IBM Case Manager extends the IBM Content Navigator JavaScript Toolkit, and it provides the following sets of JavaScript APIs:

- IBM Content Navigator Model layer API
- IBM Content Navigator Visual widget API
- IBM Case Manager Model layer API
- IBM Case Manager Page widget API

Benefits of the APIs

- Enables custom widget developers to easily extend IBM Content Navigator and IBM Case Manager functionality.
- Separation of business logic and User Interface (UI)
  - Model layer clearly partitions the business logic from the UI layer.
- Shared model objects
  - Model objects that contain business/server side objects can be shared across the default and the custom widgets
- Reusable components
  - Reuse visual widgets from the IBM Content Navigator widget library to build more complex custom UI widgets and dialogs
  - Reuse and extend IBM Case Manager page widgets for customizations.

IBM Content Navigator APIs

IBM Content Navigator JavaScript Toolkit summary

The following table lists the important IBM Content Navigator JavaScript Toolkit packages for model layer and visual widgets APIs. It also shows the important classes in the package, when to use and example scenarios.

Package	Example classes	When to use?	Example scenario
ecm.model Objects in the Content Server	Desktop Repository ContentItem WorkItem SearchTemplate SearchCriterion ResultSet	To access documents or folders (ContentItem objects) associated with a Case or a WorkItem. To retrieve information about the currently logged in user.	Use a script action to retrieve the document id of a document that is filed in a case.
ecm.widget Generic visual widgets	AddContentItemGeneralPane AddContentItemPropertiesPane AddContentItemSecurityPane ContentClassSelector	To build a custom dialog or page widget	Create a custom page widget that shows the selected document properties in the widget instead of in a dialog
ecm.widget.dialog Content operation dialogs Search dialogs Message dialogs	AddContentItemDialog CheckInDialog EditPropertiesDialog SearchBuilderDialog ErrorDialog StatusDialog	To create a custom dialog that extends an existing dialog with modifications	Create a custom Add Document dialog that allows you to select documents only from local file system.

IBM Content Navigator Model layer API

These APIs define the client-side model for IBM Content Navigator.

- The modeling classes do the following tasks:
  - Define communication with the mid-tier services
  - Provide the data server interaction APIs
  - Support caching.
- The data in the content servers is stored as model objects and used by the visual widgets.
  - The UI widgets interact with the content servers through the APIs on the model objects.

IBM Content Navigator Visual widget API

Visual widget library is based on Dojo and IDX. It provides generic User Interface widgets, panes, and dialogs for running content and workflow-related operations.

- Document and Folder-related widget components
- Workflow-related widget components
- User-related widget components
- A number of dialogs for document, search and workflow operations that can be used as is or extended.

Note

For this unit, you use only model API for the Script Adapters.

IBM Case Manager JavaScript API

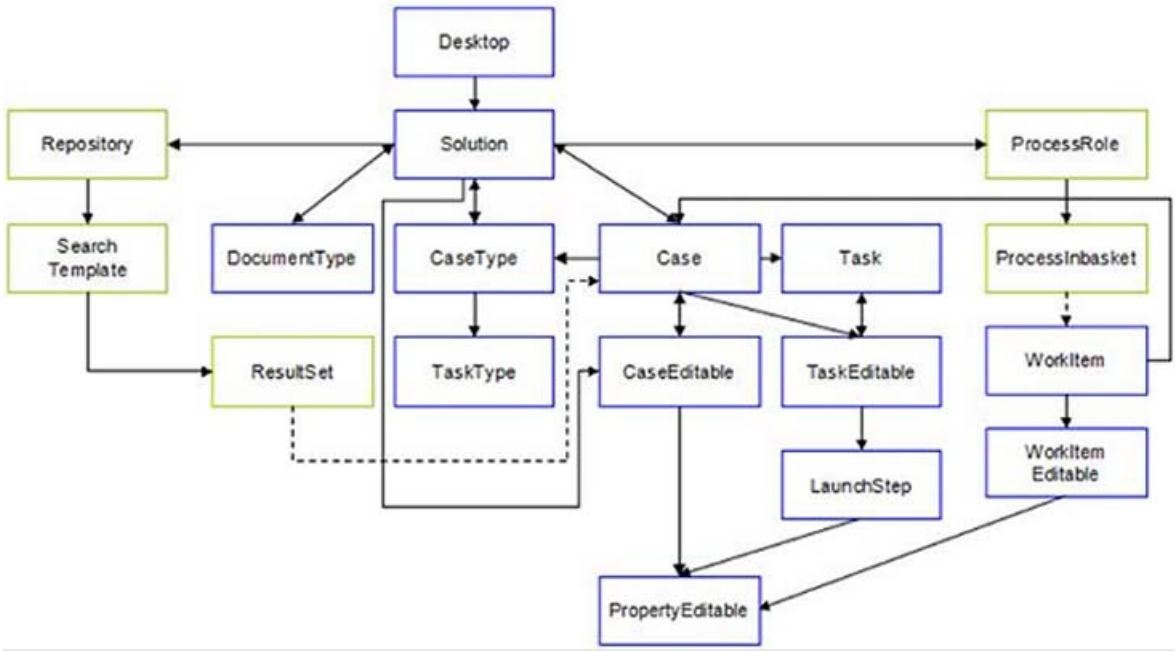
IBM Case Manager Model API

The Model API are JavaScript classes that are part of the larger Case Manager JavaScript Toolkit.

- They are non-UI components and represent Case Manager objects in the repository such as Solution, Case, and Task.
- They communicate with mid-tier services through a navigator plug-in.
- Similar to the navigator model API and services, the public API is the JavaScript Model API. The plug-in services are not public API.
  - Calling the plug-in services directly through HTTP by custom code is not supported.
  - The only supported use is a JavaScript client that calls the model classes.
- They extend and reuse some navigator model classes.
- They provide functions where it is not sufficient or possible to achieve through the navigator API alone.

IBM Case Manager Model layer objects

IBM Case Manager extends the IBM Content Navigator and you can retrieve information about the Solutions from the IBM Content Navigator Desktop object.



- The boxes that are represented in green are IBM Content Navigator objects.

IBM Case Manager Model API usage

The Model API can be used in the following scenarios.

- To develop a custom widget
  - Model object is passed as part of the payload of an event
- To use in a Script Adapter

IBM Case Manager Model API for Solutions

- Retrieve the deployed solutions
  - Solutions that are deployed to any target object store registered with the navigator desktop
- For a particular solution, retrieve
  - The case type or documents types
  - Information about the properties across all case types
  - The roles
  - The static pages to show for a particular role

IBM Case Manager Model API for a case type

- For a particular case type
  - Attributes include
  - What rights the user has cases of that case type
  - Whether dynamic (custom) tasks are supported
  - The default view definition to use when editing cases of this type
- Retrieve information about the properties of the case type
- Retrieve the discretionary task types that can be created
- Discover the particular page to use for a page type and role
- Search for dynamic tasks compatible with this case type (attached to cases of this type)

IBM Case Manager Model API for cases

- Retrieve a particular case or obtain a Case model object given a navigator ContentItem (for example from a folder-based search)
- Create a case
- Update the properties of a case
- Split a case

- Relate cases together
- Retrieve the cases that are related to a case (through splitting a case or generally relating cases together)
- Add and retrieve comments on cases
- Retrieve the case history
- Retrieve the tasks that are associated with a case
- Classes for the new case timeline history

IBM Case Manager Model API for tasks

- For task objects
  - Start a manual task
  - Enable and disable certain tasks
  - Stop and restart the workflow that is associated with a task
  - Create a discretionary task
- Dynamic (custom) tasks
  - Create a dynamic task
  - Update the data that describes the dynamic task and indicate if the data is valid (allowing the task to be started)
  - Start a dynamic task, causing dynamic workflow to be created and started

IBM Case Manager Model API for work items

Case Manager model shows a WorkItem object that encapsulates Case Manager specific behavior such as better integration with case properties in the API.

- Retrieve a work item
- Lock or unlock the work item
- Save or complete the work item
- Update the step parameters (properties) when saving or completing

IBM Case Manager JavaScript Toolkit summary

The following table lists the important IBM Case Manager JavaScript Toolkit packages for model layer and visual widgets APIs. It also shows the classes in the package, when to use and example scenarios.

Package	Example classes	When to use?	Example scenario
icm.model Objects in the Case Manager system. No UI components	Case CaseComment CaseRelationship HistoryEvent PropertyController Solution Task	- To access ICM data when you create custom widgets and actions. These objects are used with Content Navigator model. - To access data for scripting events.	- Get the case identifier for the case displayed on case detail page. - Get editable model object. - Update a case property.
icm.action The default actions that are provided by Case Manager.	AddCustomTask SendLink ShowLink, AddDocumentfromLocal	To build a custom action: You can wrap an existing ICM action or create a new custom action.	Create a custom action for case document that publishes an event
icm.base Base classes that are used to create custom page widgets and actions	BasePageWidget BaseActionContext Constants WidgetAttributes _EventStub	When creating a custom widget or action, use these base classes to provide the infrastructure and fill in the implementation with custom behavior	Create a custom search widget Create an action to display data from an external system.
icm.dialog ICM-provided dialogs	AddCommentDialog, AddTaskDialog, DynamicTaskEditorDialog	To add a dialog to a custom widget or an action	Display dialog to add a comment when user clicks an icon in a content list
icm.pgwidget Classes that represent the page widgets	Attachment, CaseForm, CaseInfo, CaseList, CaseSearch, CaseToolbar, CaseVisualizer	To create a custom widget that includes a page widget To extend a page widget with more behavior	Extend the in-basket widget to change font to red for overdue items Add new tab in Case Info
icm.util Utility classes	Coordination, SearchPayload, In-basketFilterUtil	To participate in processing a page, such as dispatching a work item or saving a case. To build nested CE queries for flexible search.	Custom widget that saves data to an external system can hook into the dispatch of a work item
icm.widget.menu Toolbar and pop-up menu classes	ContextualMenu, Menu, MenuManager, Toolbar	To include a context menu or toolbar in a custom widget, and use the Page Layout Designer to enable configuring the menus	Implement retrieval of data from an external system with a dialog that is configured as a toolbar action in a custom widget

Collaborative editing of objects

IBM Case Manager editable objects

- Core objects in the API such as Case and WorkItem do not show scratchpad behavior.
  - State of the objects always reflects what was last persisted.
  - The behavior is similar to the Content Navigator objects such as ContentItem and ecm.model.WorkItem.
- Some Case Manager model objects show an editable object that provides scratchpad semantics.
  - Modify properties and other attributes of the object before saving.
  - Multiple widgets can share an Editable object. Each widget can apply its changes before saving.

Model objects with corresponding Editable objects

The following table lists the model objects and their corresponding editable objects. It also shows when these objects are used.

Model object	Corresponding Editable object	When is it used?
Case	CaseEditable	- To Modify case properties - To Create a new case.
WorkItem	WorkItemEditable	To modify step parameters when completing or saving a step
Task	TaskEditable / LaunchStep	Creating a discretionary task that is associated with a FileNet BPM workflow
<Property objects>	A collection of PropertyEditable objects	Represents the properties

Case object example for modifying the properties

```
var caseEditable = caseObject.createEditable()
var p = caseEditable.propertiesCollection[“SOL_StringProp”]
p.setValue(“String value”);
caseEditable.save(lang.hitch(this, this._saveComplete));
```

- When editable is saved, changes are reflected back on main non- editable.
  - Client can subscribe to change notification on editable or non-editable.

  - Client such as properties widget subscribes to change notifications at each property level
- Multiple editables can exist for a particular non-editable.
  - If any are saved, the non-editable and other editables are updated.
- Beside property value, other property attributes can change when External Data Service is involved.

Creating a new object

- A pending Editable object is obtained first.
- Main model object doesn't exist until pending Editable is saved.

```
solution.createNewCaseEditable(caseType,
   lang.hitch(this, function(caseEdit){
      var p = caseEdit.propertiesCollection[...];
      p.setValue("String value");
      caseEdit.save(lang.hitch(this, this._saveNewComplete));
});
...
_saveNewComplete: function(caseEdit) {
   var caseObj = caseEdit.caseObject;
},
```

Script Adapter widget

Characteristics of Script Adapter

- A Script Adapter is a special widget that the user can place it on a page.
- The user enters JavaScript in the widget that runs when it is called.
- Script Adapters are wired to other widgets.
- Script Adapter widgets only receive events from the other widgets to which they are wired.
  - Other widgets can receive events that are broadcast.

Script Adapter widget usage

You can use Script Adapter widget for the following tasks:

- To transform the data that is published by one widget into a different format for another widget.
  - Example: Transform the data that is published by the Properties widget into a format that is understood by a custom widget on a Case Details page.
- To insert logic between widget event communication.
  - Example: Run a custom validation on data that is published by the Properties widge 
- Debug your solution with Script Adapter widget.
  - Configure the Script Adapter widget to display source event payload at runtime for debugging.

How does the Script Adapter widget inserts a logic?

- When the Script Adapter widget receives an event from a widget to which it is wired, it displays the event details in the Received Event section.
- The Script Adapter then runs a script that transforms the data as a function with a “payload” parameter.
  - The payload is from the incoming event. 
- You can manipulate the incoming payload by implementing any type of logic in a script.
  - The value that your custom script returns is the payload of the outbound event of this Script Adapter widget.
  - The Sent Event section displays this information.
- Example: A Script Adapter receives a wired event with a payload of “test data” (a string value), which the Received Event section displays.
  - The Script Adapter has the following script:

```
alert("The value of the payload is: " + payload);
return "Event Payload: " + payload + "!";
```

  - The Sent Event section displays "Event Payload: test data!" as the payload for the outbound event.
  - Notice that your script added an exclamation mark to the incoming string.

Hint

Optional: Hide the Script Adapter widget so that it is not visible to the user.

Debug the events with Script Adapter widget

You can use a Script Adapter to view event data to debug problems with wires between two widgets.

In Page Designer, edit the settings for the Script Adapter widget to select one or both of the following options:

- Show Script Text
  - Select this option to show the text of the script when the script runs.
- Block Outbound Event
  - Select this option to prevent the Script Adapter widget from sending the outgoing event.
  - If you wired the Send event payload event to another widget, you can select this option to temporarily stop the Script Adapter widget from sending the event while you are debugging the script.

Important

Your script must contain basic JavaScript only and must be viewed as the body of one single function. You cannot use Dojo commands such as console.debug(); you must use the alert() statement to display information about the values in the script.

What is a payload?

- A payload is an object that is passed from, or received by, a widget when an event happens in the system.
  - Most payloads are JSON objects.
  - A payload can be a simple string.

Exercise 2.1.1: Use Script Adapter to customize the Case client

Introduction

In this lab, you show the number of items (documents and folders) that are attached to a case by using the Script Adapter widget and wiring it to the Case List widget on the Cases page.

This is an example of the use of Script Adapter. Many examples are presented through out this unit.

Procedures

Procedure 1: Start WebSphere Application Server (if it is not already running)

1. Click Start > All Programs > IBM WebSphere > IBM WebSphere Application Server V8.5 > Profiles > AppSrv01 > Start the server.
   - You can also use the Start Server.bat file in the WebSphere Admin folder on the desktop.
2. Wait for the Start the server page to close.

Note

For more information about “Start and stop System Components”, see the Appendix at the end of unit.

Procedure 2: Create a custom page

A solution for this lab is already created. In this procedure, you copy the default Cases page and modify it.

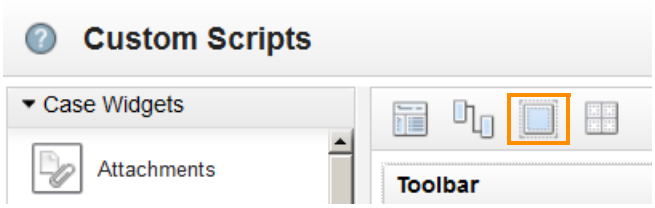
1. In Firefox, log on to IBM Case Manager Builder as an administrative user.
   - URL: `http://ecmedu01:9080/CaseBuilder`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`

2. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.
   Hover the mouse over the solution to see the links.
3. Create a custom page:
   a. Open the Pages tab and expand the Solution Pages.
   b. Click and hover the mouse over the Cases page name.
   c. Select the Copy icon on the right side of the page.
   d. In the resulting page, edit the name to `Custom Scripts` for your new page and click OK to create the copy.
   e. Save your work by clicking Save at the top of the page.
4. Leave the Pages tab opened for the next procedure.

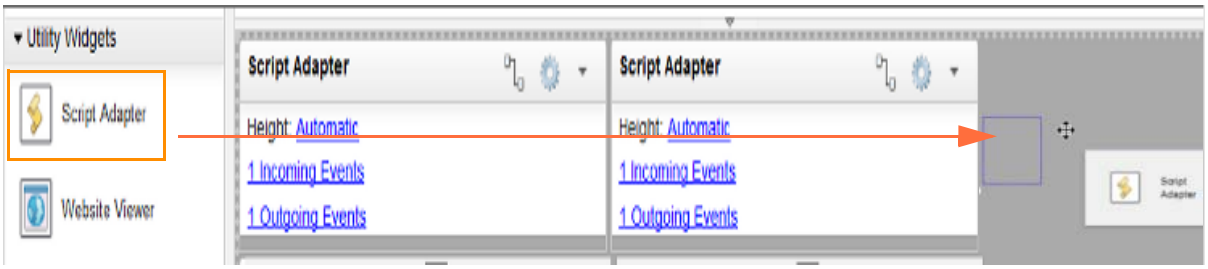
Procedure 3: Edit the page to add the Script Adapter

In this procedure, you edit the page to add the Script Adapter.

1. In the Pages tab, click your custom page (`Custom Scripts`)to edit it in Page Designer.
2. Click the Show or Hide hidden widgets button.
   a. Notice that a gray section appears on the bottom of the main layout area.
   b. The section already contains two Script Adapters.

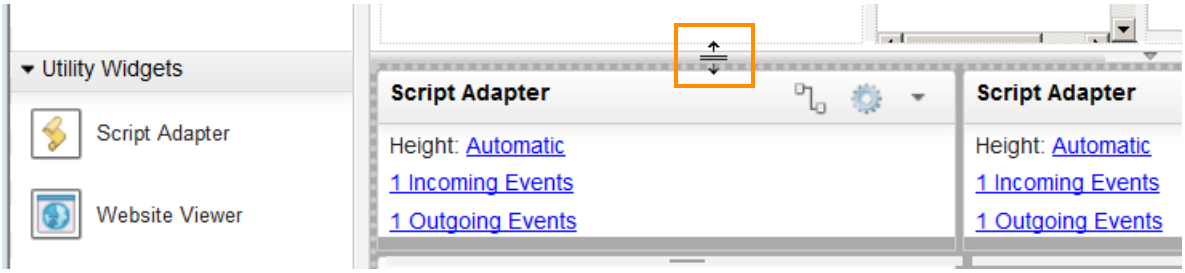


3. Drag and drop the Script Adapter widget (under the “Utility Widgets”) from widget palette on the left column to the bottom of the page on the right.



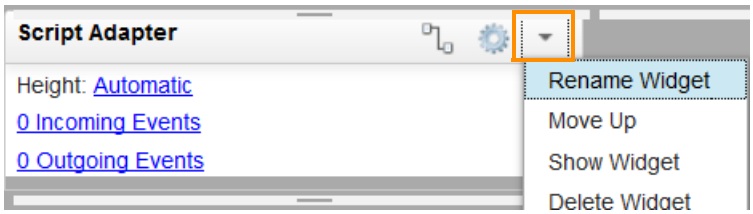
   a. The location of the widget does not matter.
   b. If your Script Adapter is not visible, scroll down the page (it might be below the existing Script Adapters).

c. You can also expand the Script Adapter area with the control that is shown in the screen capture.



4. Rename the widget

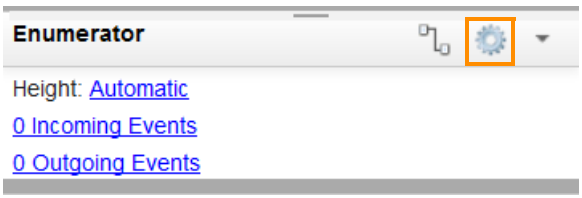
a. Click the down-arrow, and select “Rename Widget”.



b. In the “Rename Widget” dialog page, edit the Widget name as `Enumerator`.
Naming the Script Adapters helps to differentiate your widget.

c. Click OK.

5. Click the “Edit Settings” icon of the Script Adapter widget.



a. In the Script Adapter dialog window that opens, clear the text in the JavaScript text box.

6. Enter the following lines of code into the JavaScript text box.

```
var caseEditable = payload.caseEditable;
var caseObj = caseEditable.getCase();
var caseEditable = payload.caseEditable;
var caseObj = caseEditable.getCase();
var caseFolder = caseObj.caseFolder;
var actionsHandler = ecm.model.desktop.getActionsHandler();
```

```
if (actionsHandler) {
   actionsHandler.actionOpen(caseFolder, function(caseFolder,resultSet) {
      alert("The selected case folder has " + resultSet.items.length + "
      items.");
      for (var i = 0 ; i < resultSet.items.length; i ++) {
         console.log("Document ID : " + resultSet.items[i].id + "\n" +
         "Document Title: " + resultSet.items[i].name);
         alert("Document ID : " + resultSet.items[i].id + "\n" + "Document
         Title: " + resultSet.items[i].name );
      }
   });
}
return payload;
```

Hint

Optionally, copy the text from the `C:\ICM\Script Adapter Code\EnumerateDocs.txt` file and paste it.

a. Click OK to close the Script Adapter dialog page.

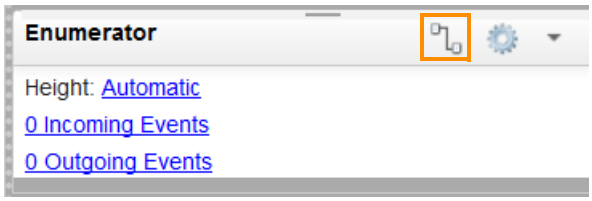
Troubleshooting

If you have any errors in the JavaScript, the OK button is disabled. Also, it shows an error message. Verify the code that you entered. If needed, clear the code, copy, and paste the code again.

How does the code works?

- The script gets case editable model objects from the payload.
- Calls the Content Navigator function `ecm.model.desktop.getActionsHandler()` to get the ActionsHandler for case folder to retrieve the contents of the folder.
- In a for loop, you then retrieve the document ID and document title.

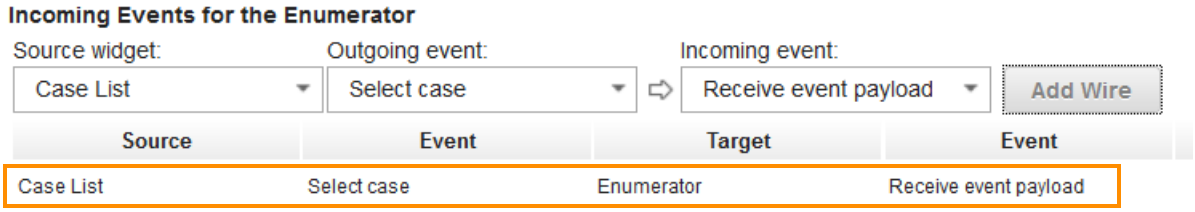
7. Click the Edit Wiring icon for the Script Adapter widget.



8. Wire the Script Adapter to the Case List widget to receive the Select Case outgoing event.
   a. In the “Wire Events” page > “Incoming Events for Enumerator” section, select the following values from the list for each field:

Field	Value
Source widget	Case List
Outgoing event	Select Case
Incoming event	Receive event payload

   b. Click Add Wire.
   c. Validate that the completed wiring looks like the following screen capture.



9. Click Save to save your work.
   a. Click OK at the bottom of the Wire Events dialog window.
   b. Click Save and then Close to close Page Designer.
   c. Leave the solution open for the next procedure.

Procedure 4: Assign the custom page to a role

1. Your solution is already opened in the Case Builder. Open the Roles tab.
2. Click the Customer Service Rep role link.
   a. Open the Pages subtab.
3. Remove the Custom Toolbar page that you created in the previous unit.
   a. Select the page, hover over, and click the Remove (trash can) icon.
4. Assign the new page that you created.
   a. Click Assign Page.
   b. Select the Custom Scripts Page.
   c. Click OK to close the dialogue window.
5. Verify that your page is listed in the Pages tab.
   a. Click OK.
   b. Click OK All to accept the changes to the role.

c. Click “Save and Close” at the top of the page to exit the solution editor.
d. Leave the Case Manager Builder open for the next procedure.

Procedure 5: Redeploy the solution

1. In the Manage Solutions page, select Lab Claims Solution and hover the mouse over.
   a. Click Commit to save the changes to the repository.
   b. Click “Commit My Changes” in the Confirmation window.
2. Select the solution again, hover the mouse over, and click Deploy.
   c. Wait for the green check mark to appear next to the solution.
3. Hover the mouse over the solution and click Test to open Case Manager Client.

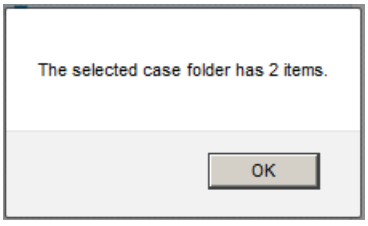
Procedure 6: Test the Script Adapter

1. In the Case Manager Client, select the Custom Scripts tab to open your custom page.
2. Search for cases.
   a. In the Search section, select Policy Family name from the list.
   b. Enter % in the next text box of the Search widget and click the Search button.
   c. Available cases are listed in the Case List widget.
3. Validate the Script Adapter widget.
   a. Select the case with the “Policy Family Name”: Beckner in the Case List widget.

Hint

If you click the Title link of a case, the case opens in the Case Details page. Select a case by clicking other columns.

   b. Verify that you get a message with number of items in that case folder. Click OK.



   c. You get another message with the Document ID, Title as you specified in the code.
   d. The code loops through the result set for each item. When you click OK on the message, if there are more items, you get more messages.

e. Close all the dialog windows.

f. Log out of the Case Manager Builder and the client and close the browser.

LESSON 2.2: Create a Case Custom Workbench Page

What this lesson is about?

This lesson describes how to create a case workbench in IBM Case Manager. From the workbench, the users view their assigned cases and the associated work.

What you should be able to do?

After completing this lesson, you should be able to:

- Create a case workbench.

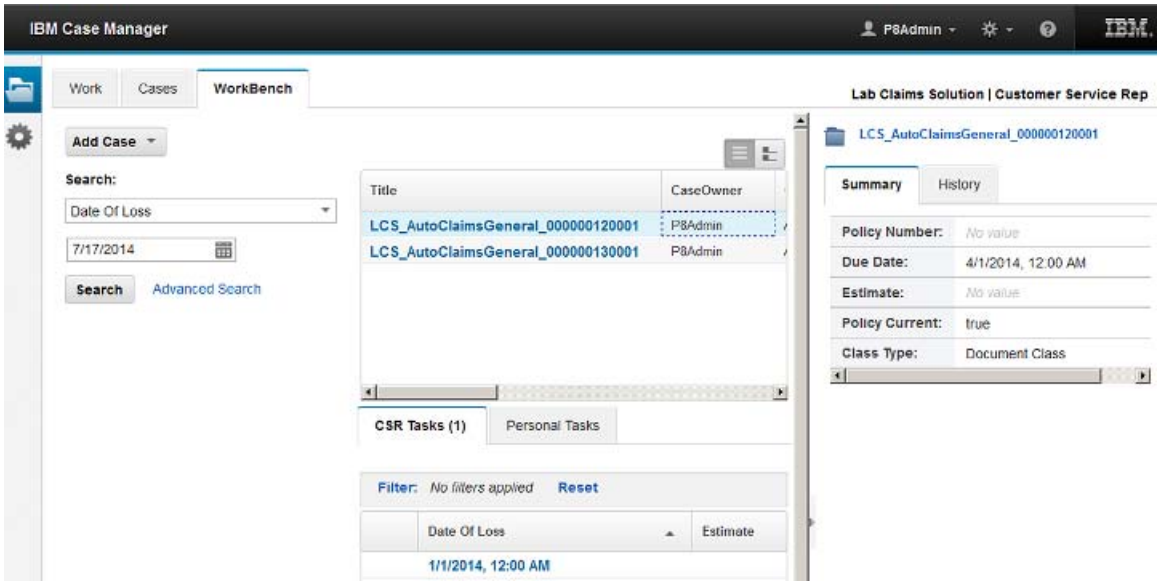
How you will check your progress?

- Hands on labs.

Case custom workbench pages

Introduction

- When a user logs in to the default IBM Case Manager Client, the client opens with two tabs initially:
  1. **Work** - The Work page shows a list of work items in the in-baskets.
  2. **Cases** - The Cases page allows users to run searches that are based on case properties.
- Work is assigned to users in Case Manager but it does not implement case ownership by default.
- The customization in this lesson provides a workitem-oriented view of case management.
  - In situations, where users own all or part of a case, you can configure Script Adapters and set up a case workbench.
  - From the workbench, the users can view only their assigned cases and the associated work.



A Case property that indicates the case ownership

The case owner has the responsibility for all of the activities that are associated with a case.

- In this lesson, you have a user who is assigned to work on a set of cases.
- A case property that identifies the case owner for a specific case.
- You use that property to determine what is shown on the custom page.

- When the Case List widget is populated, it displays only the cases for which the current user is assigned as the case owner.
- The customization in this lesson provides a workbench that is based on the association of a specific user with a case.

Note

You can create workbench pages based on any case property.

In-baskets

- An in-basket displays IBM Case Manager work items.
- Use an in-basket to view, open, and work with tasks and work items that are assigned to users.
  - If the work items are assigned to you, they are shown in your personal in-basket.
  - If the work items are assigned to a group, they are shown in one or more role in-baskets.
- The in-basket widget contains a separate tab for your personal in-basket and for each role.
- The “all assigned work in-basket” shows a list of all the open work items that are assigned to users.

In-baskets Filters

You can filter the work items list to display only the work that is assigned to a specific user or based on any other criteria.

- Define the In-baskets filter criteria in Case Manager Builder.
  - For System Properties based filters, use the Process Designer > Configuration tool.
- The filter restricts the items that are shown for faster access, and minimizes the time that you spend locating items in your inbox.
  - Example: filter for high priority items

Widgets that are used for this lesson

Introduction

The IBM Case Manager user interface is based on the IBM Content Navigator platform. In this lesson, you reuse the default page widgets; you also add code to the generic Script Adapter (SA) widgets to create a customized workbench page for the Case Manager user.

Script Adapters

- **Filter In-basket SA**
  - This widget filters only the work items that are specific for the selected case.
- **Filter Search SA**
  - This widget builds a custom search query so that active cases only for the current user are presented.
  - You can modify this sample for an administrative case view that shows the active cases for a list of specific users instead of the currently logged in user.

Default widgets that are available in the Case Manager Client

- **Case List**
- **In-Baskets**
  - The In-Basket widget lists the active work items. The Filter In-basket SA is used to ensure that only work items for the selected case are visible.
- **Search widget**
  - This widget is optional. It allows the users to further filter the cases that are listed in the Case List widget.
  - Without this widget, the users select only the cases from the Case List widget.
  - The Search widget is visible by default.

How the page is presented?

- When the user first selects the page, only the Case List widget and the Case Search widget are visible.
- When the user selects a case, the Case Summary widget and the In-basket widget are presented.
  - The case summary displays the summary information about the selected case.
  - The In-basket widget displays the active steps that are assigned to users for that case.

Filter In-basket SA - Script Adapter

Introduction

- This widget filters only the work items that are specific for the selected case.
- It is wired to the In-basket widget and
  - In-basket widget shows the work items that are filtered.

Filter In-basket SA wiring and how it works

- Filter In-basket SA is wired to the Case List and In-basket widgets.
  - The custom widget receives the “Select Case” or the “In-basket selected” outgoing event and the incoming event payload.
  - When the user selects a case in the Case List widget, a payload is sent to Filter In-baskets SA that includes the GUID of the selected case.
- The script that you add to the custom widget filters the work items from the payload that is based on the filter criteria.
  - This widget then sends the event payload (containing only the filtered work items) back to the In-basket widget.
  - This widget also causes the In-basket widget to refresh the work items list.

Note

In-basket filters can be created for any case property. Example: Work items based on user ID.

How does the script works

You create an in-basket dynamic filter in your script.

- Include the In-baskets names, that you use it for your custom page, in an array in JSON format.

```
var inbasketNames = [ {
    "queueName" : "Inbox",
    "inbasketName" : solutionPrefix + "_" + "All Assigned Work"
}, {
    "queueName" : solutionPrefix + "_" + "CustomerServiceRep",
    "inbasketName" : "CSR Tasks"
} ];
```

- Construct the data to run the dynamic filter.
  - Loop through the array of In-baskets names.
  - Add the queryFilter details.

```
for (var i = 0; i < inbasketNames.length; i++) {
    var data = {
        "queueName" : inbasketNames[i].queueName,
        "inbasketName" : inbasketNames[i].inbasketName,
        "hideFilterUI" : false,
        "queryFilter" : "(F_CaseFolder = :A)",
        "queryFields" : [ {
            "name" : "F_CaseFolder",
            "type" : "xs:guid",
            "value" : "{" + payload.caseEditable.id + "}"
        } ],
        "hideLockedByOther" : false
    };
```

- Pass the data to the `icm.model.InbasketDynamicFilter.fromJSON(data);` function to run the dynamic filter.
  - Push model data that the function returns to a model array.
  - Model array that is returned is sent out in the outgoing event payload.

```
    var model = icm.model.InbasketDynamicFilter.fromJSON(data);
    console.debug(this.name, "model:", model);
    modelArray.push(model);
    console.debug(this.name, "modelArray:", modelArray);
}
console.debug(this.name, "Returning the filter");
return {
    "dynamicFilters" : modelArray
};
```

Filter Search SA - Script Adapter

Introduction

- This widget filters only the cases that have the current user as the case owner.
  - It also filters only the active cases.
- The Case Type that you use has a case property that identifies the Case Owner.

Filter Search SA wiring and how it works

- Filter Search SA is wired to the Page Container.
- When the user opens the custom Solution page (WorkBench), the initial payload is sent to the Filter Search SA widget.
  - The Filter Search SA widget processes the initial payload, builds the search criteria (<Case Owner> property value = Current user and Case State = Active), and searches for the cases.
- The event is broadcast and received by the Case List widget.
  - This filter (with the search criteria) ensures that only the active cases for which the current user is the Case Owner are displayed in the CaseList widget.
  - The Filter Search SA also filters for a particular Case Type that is specified in the SA script. The configuration block must be configured for each Case Type.
- Optionally, you can also wire your widget to the default Search widget to further filter the cases.

How does the script works

You create and run a custom search query that is based on the case owner in your script.

- Identify the specific case type and the case property that specifies the case owner.

```
/* Identify the specific case type without the prefix. */
var myCaseType = "AutoClaimsGeneral";

/*
* Specify the case property (without the prefix)
* that is matched against the logged in user ID.
*/
var myCaseProperty = "CaseOwner";

/* The display name of the case property
var myCasePropertyName = "CaseOwner";
```

- Retrieve the logged in user and the solution.

```
/* ID of the logged in user that will be inserted into the search criteria.  */
var userId = ecm.model.desktop.userId;

/* The object for the solution we are working.  */
var solution = this.solution;

/* The solution prefix.  */
var prefix = solution.getPrefix();
```

- Build the search criterion for active cases.

```
/* Criterion for the search we are building. ( display only working cases.)  */
var criterion = new ecm.model.SearchCriterion({
    "id" : "cmAcmCaseState",
    "name" : "CaseState",
    "selectedOperator" : "EQUAL",
    "dataType" : "xs:integer",
    "defaultValue" : 2,
    "value" : 2
});
```

- Build the search criterion for user ID.

```
var criterion1 = new ecm.model.SearchCriterion({
    "id" : myCaseProperty,
    "name" : myCasePropertyName,
    "selectedOperator" : "EQUAL",
        "dataType" : "xs:string",
    "defaultValue" : userId,
    "value" : userId
});
```

- Set up the search parameters.
  - Add the object store, solution name, case type, and search criteria to a variable.

```
/* Set object store Id on params.  */

params.ObjectStore = solution.getTargetOS().id;

/*  Set up the search parameters on params.  */

params.criterions = [ criterion, criterion1 ];
params.CaseType = caseType;
params.solution = solution;
```

- Create an instance of a search payload object and pass the parameters.
  - Call the getSearchPayload function.
  - Within the function, call the onBroadcastEvent method.

```
var searchPayload = new icm.util.SearchPayload();
searchPayload.setModel(params);

searchPayload.getSearchPayload(dojo.hitch(self, function(payload) {

self.onBroadcastEvent("icm.SearchCases", payload);
```

Preferred practices

Firefox browser

- Firefox is preferred as a browser to work with IBM Case Manager.
- If you use other browsers, you must take into account that they might interpret the JavaScript differently.
- Firebug is a helpful debugging tool when you are modifying the JavaScript.
  - It runs only in Firefox.
  - You must use a compatible debugging tool for other browsers.

Create new pages for customization

- When you customize a page, create a page by copying a working page.
- You then modify the new page.
- This approach allows you to revert to the default configuration with minimal effort

Script Adapter naming

Append SA (or any other label) to the end of your Script Adapter name to make it easier to find it in the console log.

Lab overview

To create a Case workbench page, do the following steps:

1. Configure a filter for the `F_CaseFolder` system field of the in-basket.
2. Create a Case property that indicates the case ownership, and assign it to a Case Type.
3. Copy the default Cases Solution page to create the workbench page.
4. Edit the new workbench page.
   a. Add In-baskets widget to the new page.
   b. Disable Search event broadcasting.
   c. Create Filter In-baskets SA Script Adapter.
   d. Create Filter Search SA Script Adapter.
   e. Wire the Script Adapters.
   f. Wire Case List widget to In-baskets to trigger refresh.
5. Assign the page to the roles where it is used.
6. Save and Deploy the solution.
7. Test the workbench page.

Demonstrations

Create a Case Custom Workbench Page

Click here to watch the demonstration.

Exercise 2.2.1: Configure your system for the workbench page

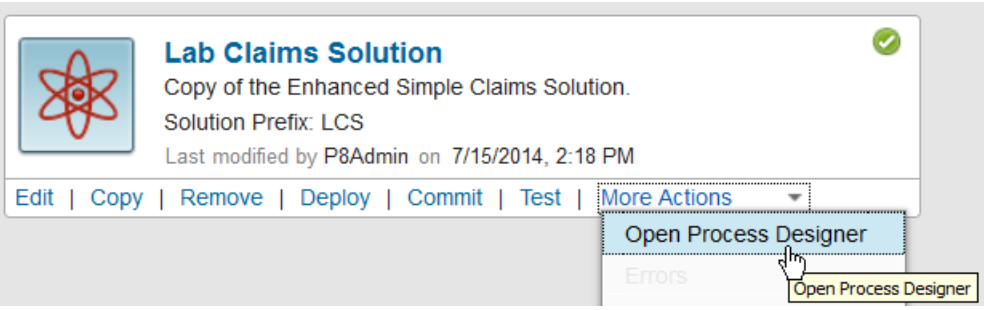
Introduction

In this lab exercise, you configure your Case Manager system for the workbench page. You create filters for the In-baskets that you use in your custom page. You add a case property for the Case Type that you use to create the cases. This property value indicates the case owner.

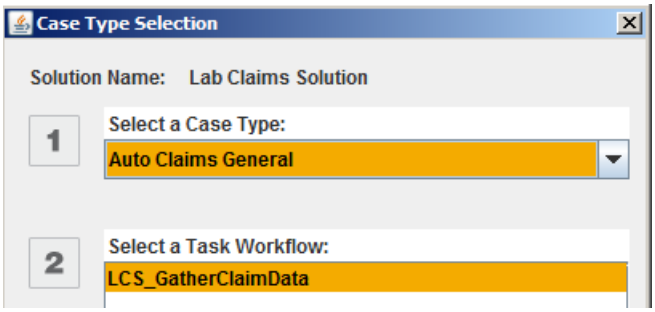
Procedures

Procedure 1: Configure queues to add a filter

1. In Firefox, log on to IBM Case Manager Builder as an administrative user.
   - URL: `http://ecmedu01:9080/CaseBuilder`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Start the Process Designer from the Solution.
   a. Hover over your solution, click More Actions, and then click “Open Process Designer”.

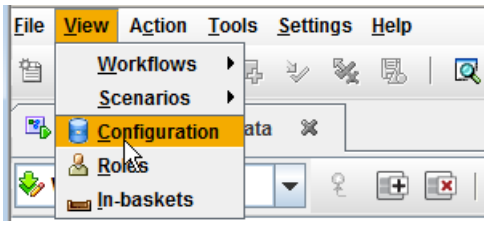


3. In the Process Designer tool, select the case type and the Task Workflow and click OK.

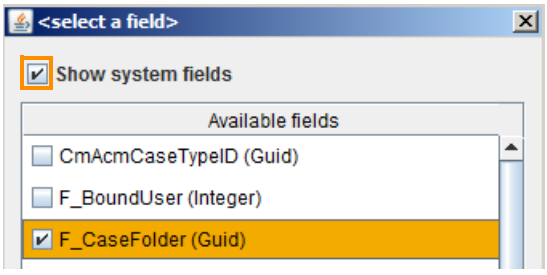


The Workflow is opened in the Process Designer.

4. Click View >Configuration. The Configuration tab is opened.

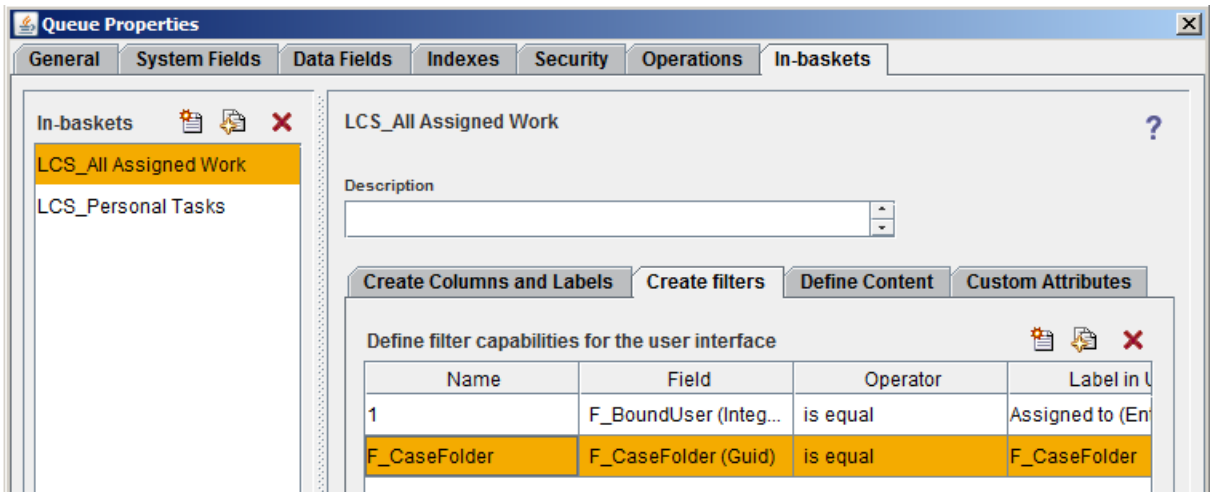


5. Open the In-baskets configuration tab for User Queues:
   a. Expand User Queues > Inbox
   b. Right-click Inbox and select properties.
   c. In the Queue Properties window, select the In-baskets tab.
   d. Select the “LCS_All Assigned Work” in-basket in the left pane.
6. Configure a filter for the in-basket:
   a. Select the “Create filters” tab in the right pane and click the Add icon. 
   b. A new row is added.
   c. In the NewFilter2 row > Field column, click and then select the ellipsis (**...**) button to display the “Available Fields” menu.
   d. Select “Show system fields”. A list of system fields are added to the menu.
   e. Select `F_CaseFolder (Guid)`from the list and then click OK.



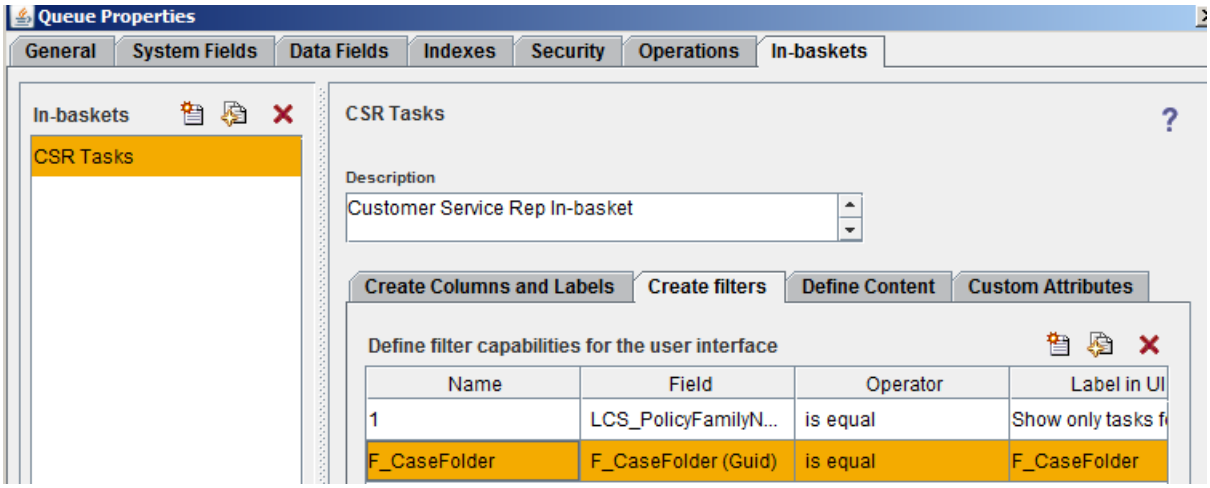
f. To rename the filter, double-click in the Name column. Clear the text, and type `F_CaseFolder`.
g. The “Label in UI” column updates automatically to match the Name column.
h. If the UI column is not updated, then double-click in that column and type `F_CaseFolder`.
i. Click OK.

7. The completed configuration must look like the following screen capture.



a. Click File > Solution > Save to save the changes.

8. Open the In-baskets configuration tab for Work Queues:
a. Expand Work Queues. Right-click LCS_CustomerServiceRep and select properties.
b. In the Queue Properties window, select the In-baskets tab.
c. Select the “CSR Tasks” in-basket in the left pane.
d. Repeat Step 6 to configure a filter for this In-basket.



9. Click File > Solution > “Save and Close” to save the changes and close the tool.

Troubleshooting

What to do when you are configuring the queues, if the pages stops responding? It might be due to inactivity, and the session expires. Close the browsers, log in to the Case Manager Builder again, and reopen the configuration tool.

Note

You must edit all the queues that you plan to show on your Workbench page. When you edit the queue configuration, it applies to all case types for the solution because the role and queue configuration is shared across the entire solution.

10. In Case Manager Builder, select Lab Claims Solution and commit the changes.
11. Leave the Case Manager Builder opened for the next procedure.

Procedure 2: Add a case property to assign a case owner

1. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.
2. Add a case property.
   a. In the Properties tab, click Add Property and select New from the list.
   b. Enter `CaseOwner` for the Name field, leave the default (String) for the Type field.
   c. Optionally, enter a description.
   d. Click OK.
3. Add the CaseOwner property to the Auto Claims General case type.
   a. Select the Case Types tab.
   b. Click the Auto Claims General link.
   c. In the Case Type page, select Properties from the left pane.
   d. Click Add Property > Existing, select `CaseOwner` and click OK.
   e. Click OK All.
   f. Verify that CaseOwner is added to the list.
4. Click “Save and Close”.
5. Optionally, commit the changes to your solution.
6. Leave the Case Manager Builder opened for the next procedure.

Exercise 2.2.2: Create a case custom workbench page

Introduction

In this lab, you create a custom workbench page to display cases that are assigned for a specific user and to display the associated work items. To achieve this configuration, you create a custom page that contains the default widgets and Script Adapters.

Procedures

Procedure 1: Create a page

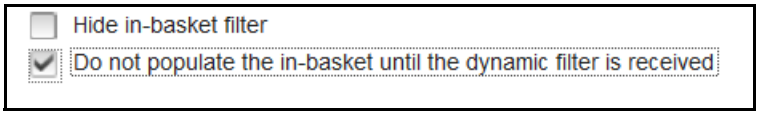
The workbench page is a customized version of the default Cases page. In this procedure, you create a copy the default Cases page layout for your workbench page.

1. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.
2. Create a custom page:
   a. Open the Pages tab and expand the Solution Pages.
   b. Hover the mouse over the Cases page name.
   c. Select the copy icon on the right side of the page.
   d. In the resulting page, edit the name to WorkBench for your new page and click OK to create the copy.
   e. Save your work by clicking Save at the top of the page.
3. Leave the Pages tab opened for the next procedure.

Procedure 2: Customize your page

1. In the Pages tab, click your custom page (WorkBench)to edit it in Page Designer.
2. Reduce the Height of the Case List widget to make space for the In-baskets widget.
   a. Click the “100%” link for the Height field.
   b. In the Resize dialog window, edit the value to 250 pixels.
   c. Click OK.

3. Drag the In-baskets widget from the Case Widgets pallet on the left pane, to your page right below the Case List widget.
   a. Click the “Edit Settings” icon for the In-baskets widget.
   b. Select the “Do not populate the in-basket until the dynamic filter is received” option.



   c. Click OK.
   d. Click Save, and then Close to close the Page Designer.

Note

Enabling this option delays the in-baskets display until the dynamic filter is received. You configure the filter in a later procedure in this lesson.

4. Save your work by clicking “Save and Close”.
   a. In the Manage Solutions page, commit the changes to your solution.
5. Leave the Case Manager Builder opened for the next lab exercise.

Exercise 2.2.3: Add a Script Adapter to filter In-baskets

Introduction

In this lab, you add a Script Adapter to your custom workbench page to filter the information that is presented in user In-baskets. When you select a case, you see only the work items that are associated with the selected case.

For this lab, to display the Activity work items for a selected case, you filter the in-baskets that are based on the case folder GUID. You already configured this filter in Process Designer in a previous procedure.

Procedures

Hint

For screen captures of the steps that are detailed in the following procedures, refer to Procedure 3: Edit the page to add the Script Adapter, page. 2-16 in the previous lesson.

Procedure 1: Create Filter In-Basket SA Widget

1. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.
2. In the Pages tab, click your custom page (`workBench`)to edit it in Page Designer.
3. Click the Show or Hide hidden widgets button.
   a. Notice that a gray section appears on the bottom of the main layout area.
   b. The section already contains two Script Adapters.
4. Drag and drop the Script Adapter widget (under the “Utility Widgets”) from widget palette on the left column to the bottom of the page on the right.
   a. If your Script Adapter is not visible, scroll down the bottom section (it might be below the existing Script Adapters).
   b. You can also expand the Script Adapter area.
5. Rename the widget:
   a. Click the down-arrow, and select “Rename Widget”.
   b. In the “Rename Widget” dialog page, edit the widget name as `Filter In-Basket SA`.

c. Click OK.

6. Click the “Edit Settings” icon of the Filter In-Basket SA widget.
   a. In the Script Adapter dialog window that opens, clear the text in the JavaScript text box.
   b. Copy the code from the `C:\ICM\Script Adapter Code\In-Basket SA.txt` file and paste it into the JavaScript text box.
   c. Click OK to close the Script Adapter dialog page.

Troubleshooting

If you have any errors in the JavaScript, the OK button is disabled. Also, it shows an error message. Verify the code that you entered. If needed, clear the code, copy, and paste the code again.

7. Save the changes to the solution by clicking Save at the top of the page.
8. Leave the Page Designer open for the next procedure.

Procedure 2: Wire the Filter In-Basket SA widget

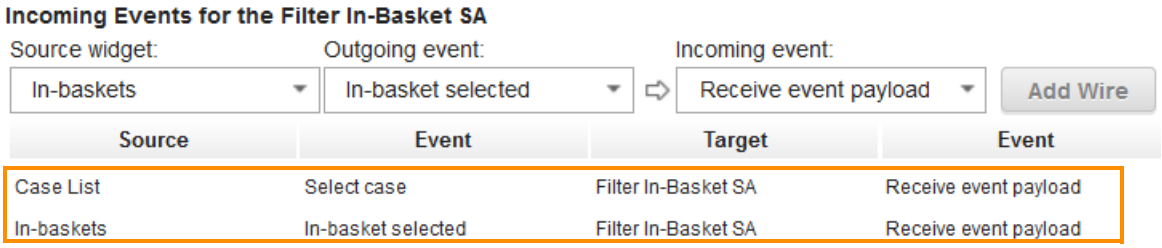
1. the Page Designer, click the Edit Wiring icon for your Script Adapter widget.
   a. The “Wire Events” page opens.
2. In the “Incoming Events for Filter In-Basket SA” section, complete the following steps:
   a. Select the values in the following table and click Add Wire.

Field	Value
Source widget	Case List
Outgoing event	Select Case
Incoming event	Receive event payload

   b. Repeat the wiring with the values in the following table:

Field	Value
Source widget	In-baskets
Outgoing event	In-baskets selected
Incoming event	Receive event payload

   c. Validate that the completed wiring looks like the following screen capture.



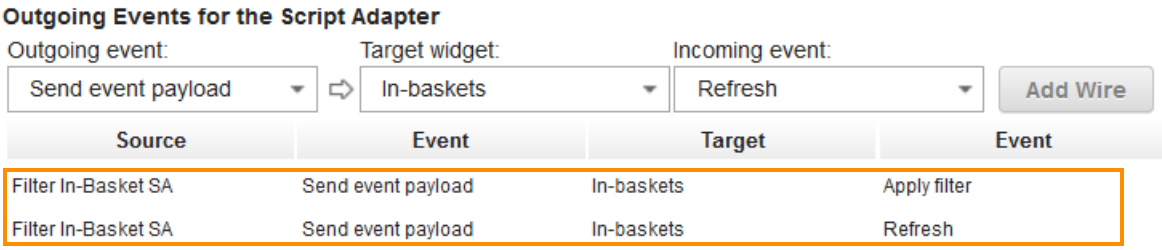
3. In the “Outgoing Events for Script Adapter” section, complete the following steps:
   a. Select the values in the following table and click Add Wire.

Field	Value
Outgoing event	Send event payload
Target widget	In-baskets
Incoming event	Apply filter

   b. Repeat the wiring with the values in the following table:

Field	Value
Outgoing event	Send event payload
Target widget	In-baskets
Incoming event	Refresh

   c. Validate that the completed wiring looks like the following screen capture.



4. Click OK at the bottom of the page to close the “Wire Events” page.
5. Save your work.
   a. Click Save and then Close to close Page Designer.
6. Leave the solution opened for the next procedure.

Procedure 3: Assign the custom page to a role

1. Your solution is already opened in the Case Manager Builder.
   a. Open the Roles tab.
2. Click the Customer Service Rep role link.
   a. Open the Pages subtab.
3. Optional: Remove the “Custom Scripts” page that you created in the previous lesson.
   a. Select the page, hover over, and click the Remove (trash can) icon.
4. Assign the new page that you created.
   a. Click Assign Page.
   b. Select Workbench.
   c. Click OK to close the dialog window.
5. Verify that your page is listed in the Pages tab.
   a. Click OK.
   b. Click OK All to accept the changes to the role.
   c. Click “Save and Close” at the top of the page to exit the solution editor.
   d. Leave the Case Manager Builder open for the next procedure.

Procedure 4: Redeploy the solution

1. In the Manage Solutions page, select Lab Claims Solution and hover the mouse over.
   a. Click Commit to save the changes to the repository.
   b. Click “Commit My Changes” in the Confirmation window.
2. Select the solution again, hover the mouse over, and click Deploy.
   c. Wait for the green check mark to appear next to the solution.
3. Hover the mouse over the solution and click Test to open Case Manager Cli 

Procedure 5: Test the Script Adapter

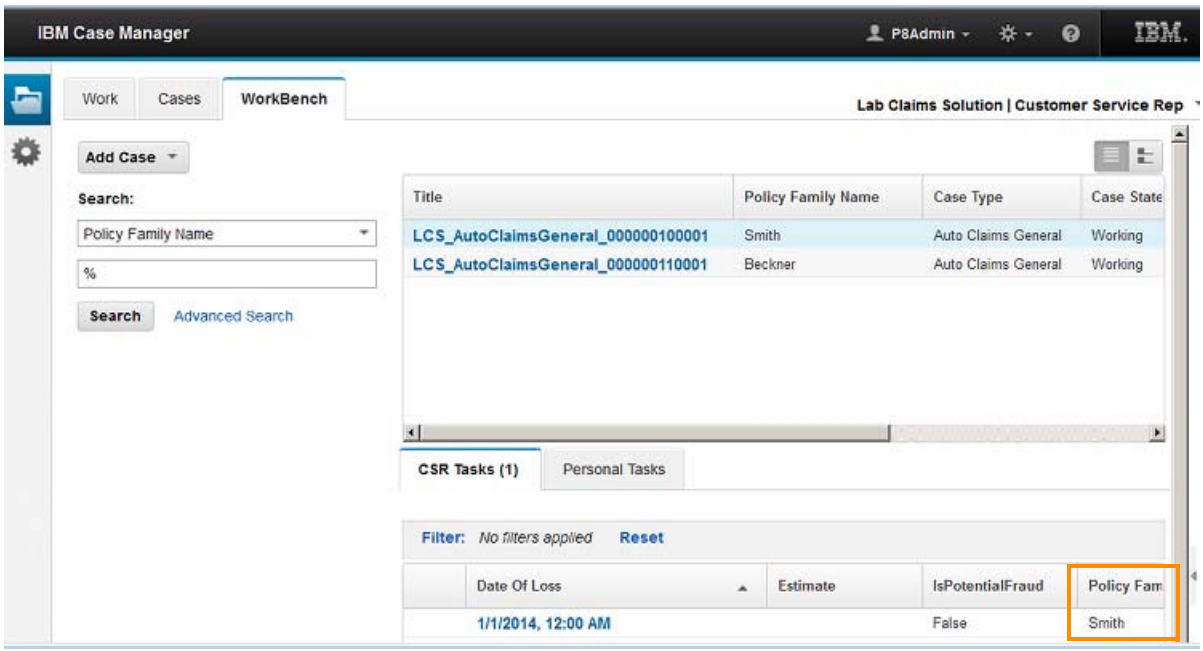
1. In the Case Manager Client, verify that Customer Service Rep role is selected on the upper right of the page.
   a. If it is not selected already, select the role from the list.
2. Click the WorkBench tab to open your custom page.
3. Search for cases.
   a. In the Search section, select Policy Family name from the list.
   b. Enter % in the next text box of the Search widget and click the Search button.
   c. Available cases are listed in the Case List widget.

4. Validate the Script Adapter widget.
   a. Select the case with the “Policy Family Name”: Smith in the Case List widget.

Hint

If you click the Title link of a case, the case opens in the Case Details page in a separate tab. Select a case by clicking other columns.

   b. Verify that the In-basket widget is shown below the Case List widget and it has a work item in the list.
   c. Maximize the browser to see the field values for the work item.
   d. Collapse the Case Information widget on the far right of the page using the arrow control.
   e. Verify that the work item has Smith as the “Policy Family Name”.



   f. To compare, open the Work tab and verify that the In-basket widget on the Work page shows all the active work items.

Note

Your custom code filtered the work items that are based on the Case Folder ID and shows only the work item for that case in the In-basket widget.

   g. Log out of the Case Manager Builder and the client and close the browser.

Exercise 2.2.4: Add a Script Adapter to filter cases

Introduction

In this lab, you add a Script Adapter to your custom workbench page to filter the cases that are presented in the Case List widget.

For this lab, you filter the cases that have the current user as the case owner. You already created a case property that is called CaseOwner for your case type in a previous procedure. You also filter only the active cases. To achieve the result, you create a custom search for the cases.

Procedures

Procedure 1: Create Filter Search SA Widget

1. In Firefox, log on to IBM Case Manager Builder as an administrative user.
   - URL: `http://ecmedu01:9080/CaseBuilder`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.

   Hover the mouse over the solution to see the links.
3. Open the workbench page:
   a. Open the Pages tab and expand the Solution Pages.
   b. Click your custom page (`WorkBench`)to edit it in Page Designer.
4. Click the Show or Hide hidden widgets button.
5. Drag the Script Adapter widget from widget palette to the bottom of the page next to Filter In-basket SA.
   a. If your Script Adapter is not visible, scroll down the bottom section.
   b. Expand the Script Adapter area.

Hint

For screen captures of the steps that are detailed in the following procedures, refer to Procedure 3: Edit the page to add the Script Adapter, page. 2-16 in the previous lesson.

6. Rename the widget:
   a. Click the down-arrow, and select “Rename Widget”.
   b. In the “Rename Widget” dialog page, edit the widget name as `Filter Search SA`.
   c. Click OK.
7. Click the “Edit Settings” icon of the Filter Search SA widget.
   a. In the Script Adapter dialog window, clear the text in the JavaScript text box.
   b. Copy the code from the `C:\ICM\Script Adapter Code\Search SA.txt` file and paste it into the JavaScript text box.
   c. Click OK to close the Script Adapter dialog page.

Troubleshooting

If you have any errors in the JavaScript, the OK button is disabled. Also, it shows an error messag  erify the code that you entered. If needed, clear the code, copy, and paste the code again.

8. Save the changes to the solution by clicking Save at the top of the page.
9. Leave the Page Designer open for the next procedure.

Procedure 2: Wire the Filter Search SA widget

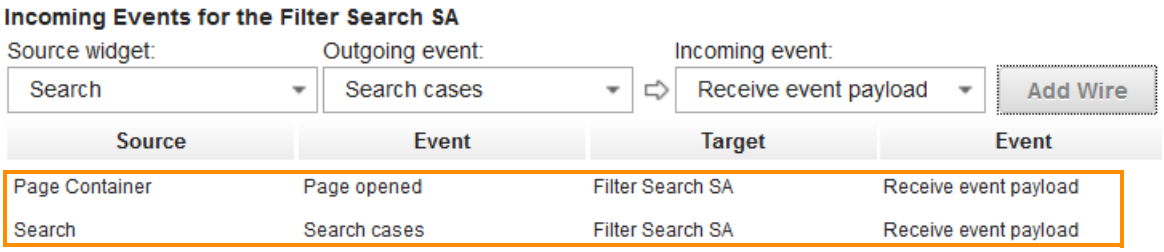
1. In the Page Designer, click the Edit Wiring icon for your Script Adapter widget.
   a. The “Wire Events” page opens.
2. In the “Incoming Events for Filter Search SA” section, complete the following steps:
   a. Select the values in the following table and click Add Wire.

Field	Value
Source widget	Page Container
Outgoing event	Page opened
Incoming event	Receive event payload

b. Repeat the wiring with the values in the following table:

Field	Value
Source widget	Search
Outgoing event	Search Cases
Incoming event	Receive event payload

c. Validate that the completed wiring looks like the following screen capture.



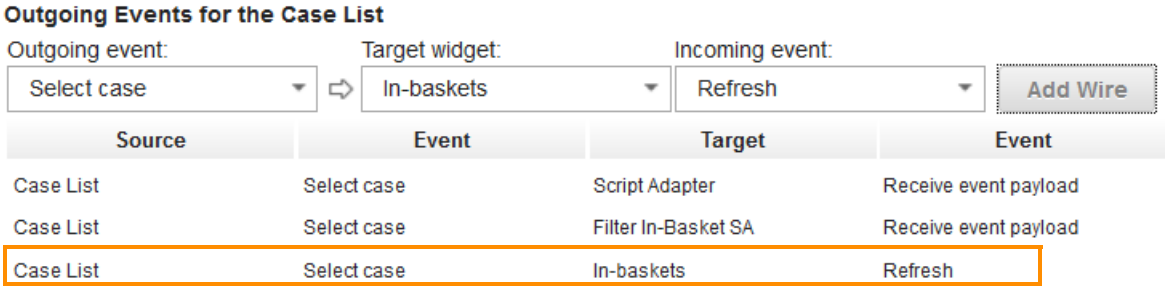
3. Click OK at the bottom of the page to close the “Wire Events” page.
4. Leave the Page Designer opened for the next procedure.

Procedure 3: Edit the Case List widget wiring

1. In the Page Designer, click the Edit Wiring icon for Case List widget.
   a. The “Wire Events” page opens.
2. In the “Outgoing Events for the Case List” section, select the values in the following table:

Field	Value
Target widget	In-baskets
Outgoing event	Select case
Incoming event	Refresh

a. Click Add Wire.
b. This wiring refreshes the In-basket when a case is selected.
c. Validate that the completed wiring looks like the following screen capture.

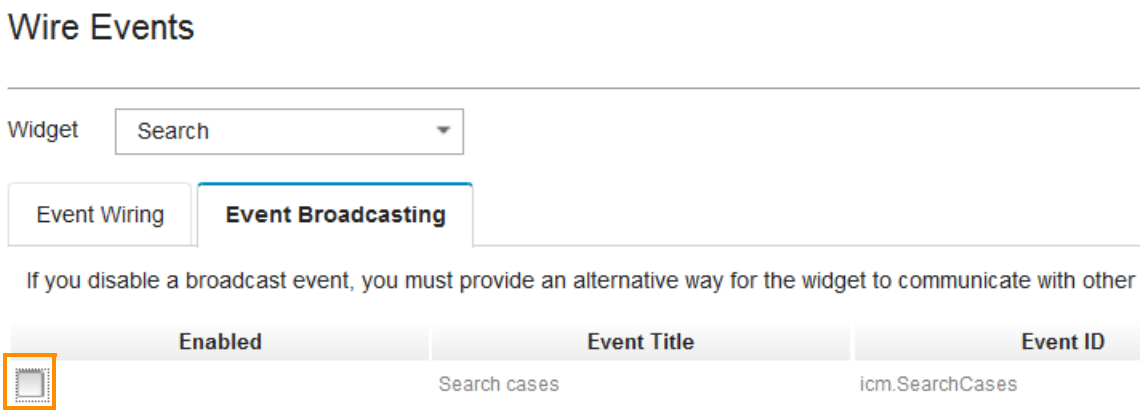


3. Click OK at the bottom of the page to close the “Wire Events” page.
4. Leave the Page Designer opened for the next procedure.

Procedure 4: Disable Broadcasting from the Search widget

For the Case List widget to receive the search payload from your custom widget, the broadcast from the default Search widget must be disabled.

1. In the Page Designer, click the Edit Wiring icon for Search widget.
   a. The “Wire Events” page opens.
   b. Select the Event Broadcasting tab.
2. Clear the Enabled option.



   a. Click OK.
3. Click Save and then Close to close Page Designer.
4. Click “Save and Close” at the top of the page to exit the solution editor.

Procedure 5: Redeploy the solution

1. In the Manage Solutions page, select Lab Claims Solution and hover the mouse over.
   a. Click Commit to save the changes to the repository.
   b. Click “Commit My Changes” in the Confirmation window.
2. Select the solution again, hover the mouse over, and click Deploy.
   c. Wait for the green check mark to appear next to the solution.
3. Hover the mouse over the solution and click Test to open Case Manager Client.

Procedure 6: Test the Script Adapter

1. In the Case Manager Client, verify that Customer Service Rep role is selected on the upper right of the page.
   a. If it is not selected already, select the role from the list.
2. Select the Cases tab to add several cases for `Auto Claims General` case type:
   a. Some cases with the CaseOwner property value as `P8Admin` and others with `clara`.
   b. Sample data for adding cases is provided in the following table. Leave the other fields that are not given in the table blank.

Important

The search criteria values are case-sensitive. Make sure that you enter P8Admin (“P” and “A” are upper cases) correctly for the custom search to work. The value “clara” has all lowercases.

Auto Claims General case type	**Case property**	**Value**
Case - 1	CaseOwner	P8Admin
	Policy Family name	Steven Heights
Case - 2	CaseOwner	P8Admin
	Policy Family name	Barbara Seth
Case - 3	CaseOwner	clara
	Policy Family name	Emma Watts
Case - 4	CaseOwner	clara
	Policy Family name	Sara Cummings

3. Refresh the browser for the pages reload since the custom widget received the initial payload when the page loads.
   a. Select the WorkBench tab to open your custom page.
4. Verify that only the cases with “P8Admin” as the case owner is listed in the Case List widget automatically without running a search. Your widget does the custom search.

Troubleshooting

If the cases are not listed, reload the page by logging out and logging back in to the Case Manager Client. The existing two cases (Policy Family Name: Smith and Beckner) are not listed because these cases were created before you added the CaseOwner property. These two cases does not have a value for the CaseOwner property.

5. You implemented the “Filter In-basket” code in the previous lab exercise.
   a. Verify that when you select a case, only that work item for that case is listed (in stead of all the active work items) in the In-basket widget.
6. Optionally, validate the other case owners.
   a. You already created a few cases with “`clara`” as the case owner.
7. Add the case owner “`clara`” as a member to the Customer Service Rep role.
   a. Open the Work tab and click Manage Roles.
   b. Select the Customer Service Rep role and click “Add Users and Groups”.
   c. Search for `clara` user and move `clara` from the Available to Selected pane.
   d. Click Add and then Save.
8. Verify the workbench page with clara as the case owner.
   a. Log out of the Case Manager Client and log in back as `clara` (password: `filenet`).
   b. Select the Workbench tab to open your custom page.
   c. Verify that only the cases with “clara” as the case owner is listed in the Case List widget.
   d. Select a case, only that work item for that case is listed in the In-basket widget.
   e. Log out of the Case Manager Builder and the client and close the browser.

3

Develop Custom Widgets

This unit provides guidance for developing IBM Case Manager custom widgets.

LESSON 3.1: Custom widget development overview

What this lesson is about

This lesson gives an overview of custom widget development environment. You create an IBM Content Navigator plug-in project in Eclipse for your custom widget.

What you should be able to do

After completing this lesson, you should be able to:

- Describe custom widget development environment.
- Create an IBM Content Navigator plug-in project in Eclipse.

How you will check your progress?

- Hands on labs.

References

IBM Case Manager 5.2 documentation

IBM Case Manager JavaScript API Reference

IBM DeveloperWorks forums

IBM Redbooks publication: Advanced Case Management with IBM Case Manager

Developing case management applications

Introduction

IBM Case Manager and the IBM FileNet P8 software provide tools for building custom web applications to manage cases that you can use in the following ways:

- Extension points and application programming interfaces (APIs) to extend Case Manager Client.
  - Add custom pages, widgets, actions, events, or services.
- The APIs to build custom applications that incorporate Case Manager features.
  - It enables caseworker to process cases without using Case Manager Client.

Note

You can use Case Manager Builder to create a solution with a template, and then modify that solution to meet your requirements. You cannot use a custom application to create a solution. You must use Case Manager Builder to create or modify solutions.

IBM Case Manager JavaScript API

IBM Case Manager provides a JavaScript API that you can use to customize your case management client application.

- The IBM Case Manager JavaScript API uses the Dojo toolkit, which is an open source JavaScript library for web development.
- In addition, you use the IBM Content Navigator JavaScript API to customize your client application.
  - It includes more modeling and widget classes that you can use in your application.

Note

The “Use Scripts to Customize Case Manager Client” unit in this course has a detailed introduction to the JavaScript APIs.

Prerequisites

To develop a page widget, you must know the following technologies, and the programming models.

- Programming models:
  - IBM Case Manager Client
  - IBM Content Navigator
- Technologies:
  - Dojo 1.8.4 (This version is packaged with IBM Content Navigator)
  - Extensible Markup Language (XML)
  - HyperText Markup Language (HTML)
  - Java 2 Enterprise Edition
  - JavaScript
  - Eclipse IDE

Creating a custom page widget and actions package

High-level steps

1. **Create an IBM Content Navigator plug-in** for the widget package.
   - The plug-in classes and files are needed to use your custom page widgets in Case Manager Client, which runs in IBM Content Navigator.
   - These files also make any custom actions available in Case Manager Client, and IBM Content Navigator.

Note

Refer to Lesson: 3.1, Custom widget development overview, on page 3-2 in this unit for more details about the IBM Content Navigator plug-in.

2. **Create the registry files** for the widget package.
   - The files are required to register the page widgets and make them available for use in Case Manager Builder.

Note

Refer to Lesson: 3.2, Create catalog and widget definition files, on page 3-17 in this unit for more details about the registry files.

3. **Implement custom widgets** and any custom actions that are used by the widgets.
   - In this step, you create the JavaScript files that implement the functions for your custom page widgets and actions.

Note

Refer to Lesson: 3.3, Implement a custom widget, on page 3-28 in this unit for more details about the implementation of custom widgets.

4. **Create the web project** for the custom page widget and actions (Optional).
   - The web project defines the folder structure that is required for building, packaging, and registering your custom widget package. Widgets are contained in an EAR file.

Note

Refer to Lesson: 3.5, Create a widget with a toolbar and a menu, on page 3-57 in this unit for more details about the web project.

5. **Package the page widgets** and actions files for deployment.
   - You create the package as a compressed file (ZIP).
6. **Deploy the custom widget package.**
   - Use the IBM Case Manager administration web client for a small widget package.
   - Use the IBM Case Manager configuration tool for a larger widget package that contains widgets in an EAR file.
7. **Add the custom page widgets to a page** in Page Designer.
8. **Save and Deploy the solution** in Case Manager Builder.
9. **Test your custom widgets** in Case Manager Client.

Note

Refer to Lesson: 3.4, Build and register a widget package, on page 3-42 and Lesson: 3.6, Build and deploy a widget as an EAR file, on page 3-76 in this unit for more details about packaging and deploying a custom widget package. Refer to Lesson: 3.7, Update an existing package with new widgets, on page 3-90 in this unit for more details about updating a widget package.

IBM Content Navigator plug-in for the custom widget package

Introduction

- In IBM Case Manager V5.2, the Case Manager Client application is deployed as a plug-in on IBM Content Navigator.
- When you create Case Manager custom widgets, you use the Content Navigator Model API.
- You must create the Content Navigator plug-in that makes your custom page widgets and actions available within IBM Content Navigator for Case Manager Client.

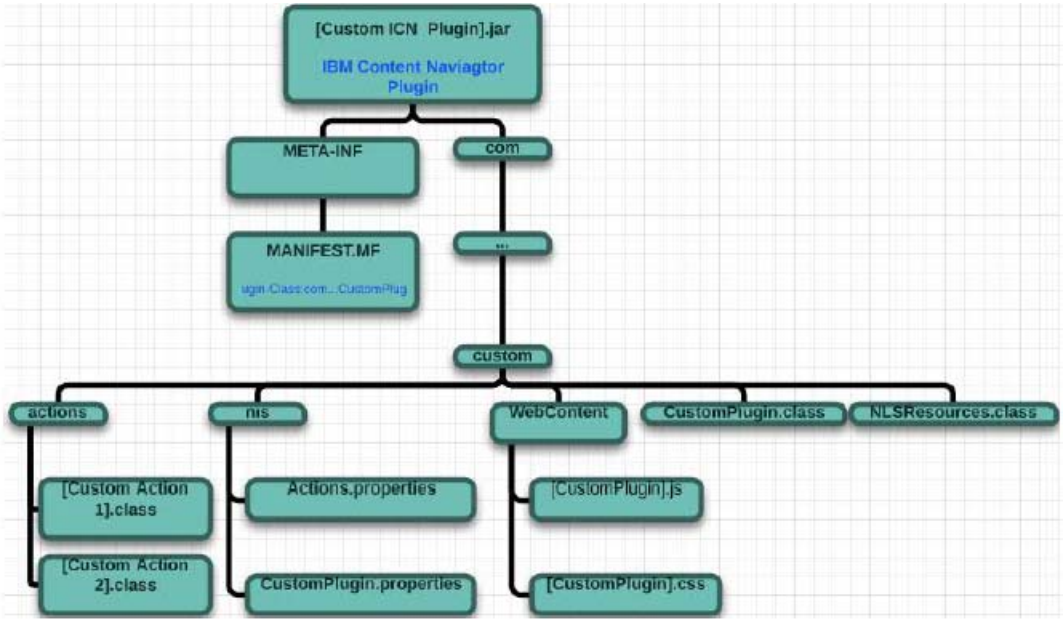
What is an IBM Content Navigator plug-in?

- A plug-in is a Java class that a developer must extend to implement a plug-in.
- It also refers to a Java Archive (JAR) file that contains Java classes to implement defined mid-tier extension points and web resources for extending the browser user interface.

Create an IBM Content Navigator plug-in

To create the plug-in package, you combine the classes and JavaScript files into a JAR file.

The following graphic shows a sample JAR file that contains the directories and files that are needed for the custom page widget and actions:



- The CustomPlugin.class describes the custom plug-in.

  - The class provides the initial JS file name for the plug-in, and the action list in the package.
- To declare the plug-in class to IBM Content Navigator, add the plug-in class to the META-INF\MANIFEST.MF.
- The WebContent\[Custom Plug-in].js file registers the Dojo module path for the custom runtime code.
- Depending on whether debug mode is on or off, the [Custom Plug-in].js file can load in the source code for debugging or in the combined and compressed code for better performance.
- Each custom action class in the actions folder also provides an action definition.
- The Java class com.ibm.icm.extension.custom.ICMCustomPlugin is the single entry point of the sample IBM Content Navigator plug-in.

Setting up the development environment for plug-ins

Development environment

You can use any Eclipse-based development environment to create IBM Content Navigator plug-ins and custom widget packages.

- IBM Rational Application Developer
- Eclipse Software Development Kit (has different packages)

Note

For the student image, you use “Eclipse IDE for Java EE Developers” package. The lab exercises in this course use the Eclipse Kepler package that is based on Eclipse Version 4.3.1.

Eclipse plug-in for IBM Content Navigator

The Eclipse plug-in for IBM Content Navigator makes the creation of new Content Navigator plug-in projects easy. It contains the extensions for Eclipse and can be integrated in your development environment.

- The plug-in contains the following Java Archive (JAR) file:
  - com.ibm.ecm.plugin.202.jar

Installation of the Eclipse plug-in

1. Copy the JAR files in the <Eclipse installation path> /dropins directory.
   - Example for base Eclipse: `C:/eclipse/dropins`
   - Example for Eclipse for WebLogic: C:/Oracle/Middleware/Oracle_Home/oepe/eclipse/plugins
   - Example for Rational Application Developer: C:/Program Files/IBM/SDP/plugins
2. Restart Eclipse.

   Sometimes it is necessary to call Eclipse with the -clean parameter from the Command Prompt to make the plug-in active: `C:/eclipse/eclipse.exe -clean`

Note

The Eclipse plug-in can be downloaded from the IBM Redbooks publication page. For the student image, the plug-ins are already included in Eclipse.

Exercise 3.1.1: Create a plug-in project in Eclipse

Introduction

In this exercise, you create an IBM Content Navigator plug-in project in Eclipse.

User accounts

Type	User ID	Password
Operating system	administrator	passw0rd
IBM Content Navigator Administrator	P8Admin	IBMFileNetP8

Passwords are always case-sensitive.

Procedures

Procedure 1: Create a plug-in project in Eclipse

Note

Eclipse IDE for Java EE Developers and the Eclipse plug-in for IBM Content Navigator are already installed on your student image.

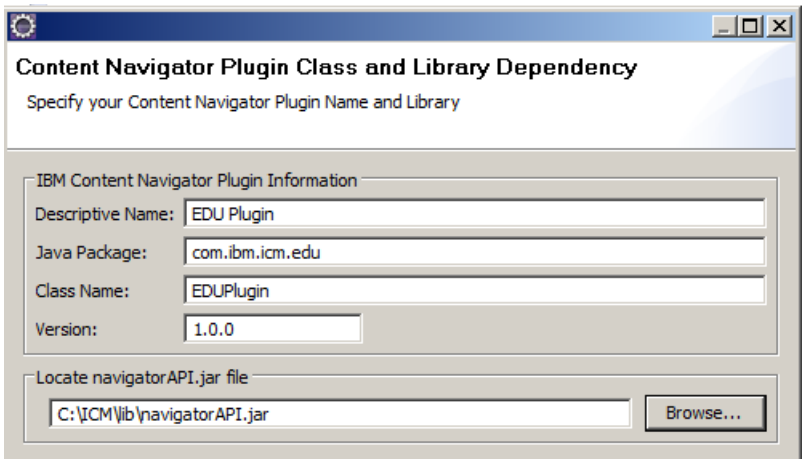
1. Open Eclipse by double-clicking the Eclipse icon in your desktop.
   a. In the Workspace Launcher page, leave the default workspace directory (`C:\ICM\workspace_Eclipse`) and click OK.
2. Open the project creation wizard.
   a. In Eclipse, click File > New > Other.
   b. In the `New` page, scroll down and select IBM Content Navigator > Content Navigator Plug-in.
   c. Click Next.
3. In the Content Navigator Plugin Project page, enter a Project Name (Example: `EDUPlugin`).

4. Leave other default settings and click Next.
5. In the “Content Navigator Plugin Class and Library Dependency” page, enter the values from the following table.

Item	Value	Notes
Descriptive Name	EDU Plugin	A string that identifies your plug-in in the IBM Content Navigator administration plug-in interface
Java Package	com.ibm.icm.edu	Namespace for your plug-in source code
Class Name	EDUPlugin	Name of your primary plug-in class. The class provides instructions to the IBM Content Navigator server about your plug-in extensions and the classes to load at run time.
Version	1.0.0	Version of the plug-in. The Version number is set in the class that is provided under Class Name.
Locate navigatorAPI.jar	C:\ICM\lib\navigatorAPI.jar	The JAR file contains the plug-in interfaces that the IBM Content Navigator provides you to build custom extensions.

Note

The navigatorAPI JAR file is installed with IBM Content Navigator under the lib directory (Example: C:\Program Files (x86)\IBM\ECMClient\lib). In your student system, this file is copied into the C:\ICN\lib folder.

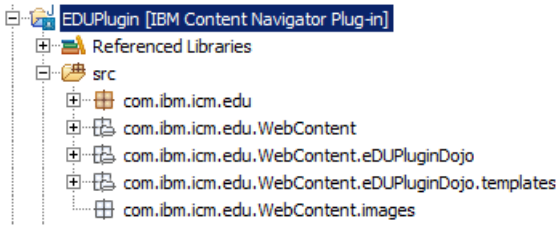


6. Click Finish.
7. If the “Remember my decision” page prompts, click No.
8. The wizard generates a working Content Navigator plug-in project.

Procedure 2: Check the packages and files

In this procedure, you check the packages and files that the wizard generates for your project.

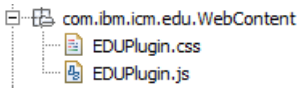
1. In the Package Explorer pane on the left, expand the `EDUPlugin [IBM Content Navigator Plug-in]` project > `src` folder.



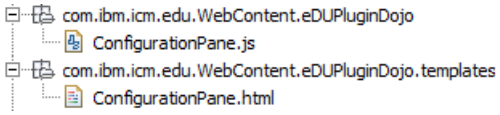
2. The following table lists the directories and their content in the src folder of your project.

Directories under the src folder of your project	Content
<PackageName> `com.ibm.icm.edu`	All the Java Classes extending the IBM Content Navigator plug-in classes
<PackageName>/WebContent	CSS files, main JavaScript plug-in, and images folder
<PackageName>/WebContent/<Dojo>	Dojo classes (where you add your page widget files)
<PackageName>/WebContent/<Dojo>/templates	HTML templates, if you choose to extend the dijit._Templated dojo class

3. Expand each directory and check the files.
   - WebContent directory (`com.ibm.icm.edu.WebContent`)

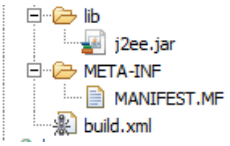


     – Root directory for your plug-in extensions to the client-side.
     – The wizard created shells that you can use to build your customizations.
     – Example: You can add a style change to the CSS file that is generated in this directory.
   - <Dojo> package (`com.ibm.icm.edu.WebContent.eDUPluginDojo`)



  - The wizard generates a Dojo package name for your custom Dojo widgets.
  - IBM Content Navigator registers this namespace for your Dojo widgets.
  - For example, if you create a custom dialog (Example: EDUDialog), you must add it to the eDUPluginDojo package and reference it as eDUPluginDojo.EDUDialog.

4. Expand the lib and META-INF folders in your project.
   - lib folder
     - It contains the j2ee.jar by default.
     - You can add the required JARs for your plug-in to the lib folder.
   - META-INF folder
     - It contains the MANIFEST.MF file that defines the main plug-in class for the project.
     - The Content Navigator plug-in registration requires this value to define the plug-in.
     - In the MANIFEST.MF file, the parameter "`Plugin-Class`" points to com.ibm.icm.edu.EDUPlugin and it is automatically set through the wizard.

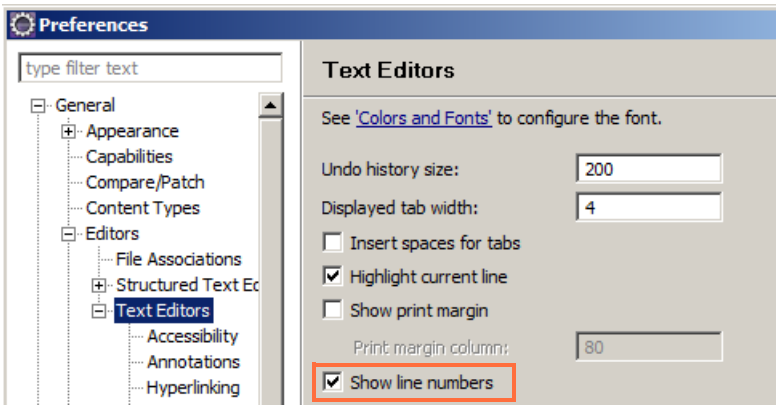


   - build.xml
     - It is an ANT script that generates a JAR file out of the existing project code.
     - This file contains the name of the output plug-in JAR file in the JAR section.

Procedure 3: Enable line numbers in Eclipse

If the line numbers are not visible in the text editor of the Eclipse, do the following steps or skip to the next procedure.

1. Select Window > Preferences from the top menu.
2. In the Preferences window, expand General > Editors, and select Text Editors.
3. Select the "Show line numbers" option.

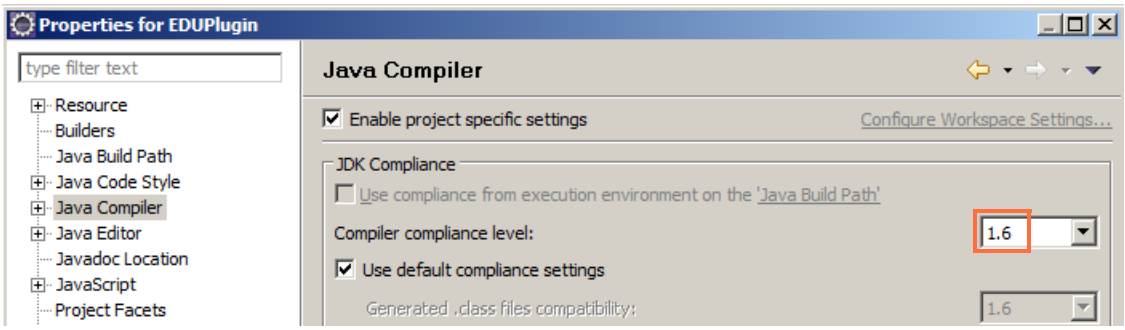


Procedure 4: Configure Java Compiler

Important

You must ensure that you are using Java Compiler Level 1.6 in your plug-in project to avoid Java Version errors during plug-in registration.

1. In Package Explorer, right-click the EDUPlugin project and select Properties from the list.
2. In the Properties for EDUPlugin page, select Java Compiler from the left pane.
3. In the right pane, select the “Enable project-specific settings” option.
4. Select the `1.6` from the list.
5. Select the Use “Default compliance settings” option and click OK.



6. When prompted, click Yes in the “Compiler Settings Changed” page to rebuild the project.

Procedure 5: Check the generated code in EDUPlugin.java file

The EDUPlugin.java file that the wizard generated is the main class for your Content Navigator plug-in.

1. If the file is not already opened:
   a. In Package Explorer, expand your project > src > com.ibm.icm.edu.
   b. Double-click the EDUPlugin.java file.
   c. Observe that the wizard generated many methods in the Java file.
   d. Close the file.
2. Your student image has a copy of the EDUPlugin.java file with only the methods that are required for this lab exercise.
   a. Replace the wizard-generated file with the lab file in the following steps.
3. Right-click the `com.ibm.icm.edu` package and click Import from the list.
   a. In the Import page, expand General, select “File System”, and click Next.
   b. In the Import > File system page, click Browse.

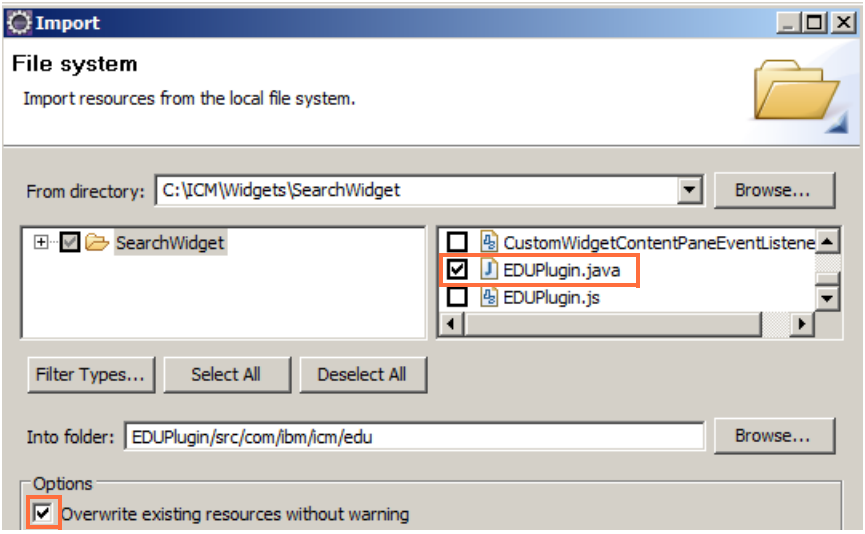
c. In the “Import from directory” page, go to C:\ICM\Widgets folder and select the SearchWidget folder.

d. Click OK.

e. Back in the Import > File system page, select EDUPlugin.java from the list.

4. Make sure that the “`Into folder`” field has the following value: `EDUPlugin/src/com/ibm/icm/edu`

a. Select the “Overwrite existing resources without warning” option.



b. Click Finish.

5. Open the EDUPlugin.java and observe the methods.

a. The id value of the plug-in is required for the Content Navigator.

b. The Name of your plug-in is displayed in the admin tool.

```
public String getId() {
    return "EDUPlugin";
}

public String getName(Locale locale) {
    return "EDU ICN Plugin";
}
```

6. The base JavaScript file is loaded when IBM Content Navigator loads the plug-in.

a. You can use this JavaScript file to apply any global changes (such as a style override) or load any JavaScript classes that must be available throughout the session.

```
public String getScript() {
    return "EDUPlugin.js";
}
```

7. The Dojo package where you are going to add your custom widget files.
   a. IBM Content Navigator registers this namespace for your Dojo widgets.
   b. Your custom page widgets become available in Case Manager Client, which runs in IBM Content Navigator.

```
public String getDojoModule() {
    return "eDUPluginDojo";
}
```

8. Leave Eclipse open through out this unit.

Note

You created an IBM Content Navigator plug-in project in this exercise. In the following lesson, you add the custom widget files to this project.

LESSON 3.2: Create catalog and widget definition files

What this lesson is about

This lesson provides guidance for creating the catalog and widget definition files. The files are used to register a custom widget package with Case Manager.

What you should be able to do

After completing this lesson, you should be able to:

- Create a catalog JSON file.
- Create a widget definition JSON file.

How you will check your progress?

- Hands on labs.

Create registry files for the custom widget package

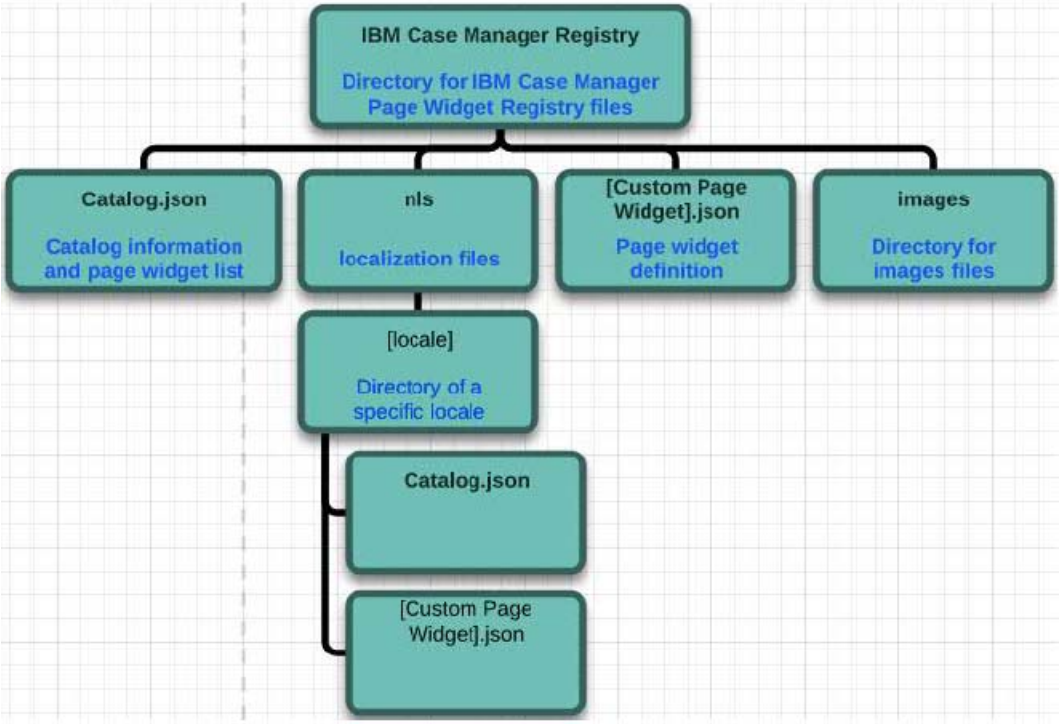
Introduction

As part of the widget package, you create files that are used to register your custom page widgets in Page Designer. After registration of the page widgets, they are available in the Page Designer palette.

- Following are the two files that are required for the registration.
  - Catalog.json
  - [Custom page widget].json
- You can also include the auxiliary files.
  - Example: Localized resource files and the image files that are used for the page widget.

Folder structure for registry files

The diagram shows a sample of the folder structure for registry files.



Catalog.json file

Introduction

The Catalog.json file provides the following information:

- A description of your custom page widget package
- A list of page widgets in the package.

Description of the custom widget package

Description of the custom widget package contains the following properties.

Property	Required?	Type	Notes
Name	Required	String	- Name for your custom page widget package. - Must specify a unique name
Description	Required	String	
Locale	Required	String	The code that identifies the locale for the current catalog. This code must correspond to the subfolder name in the ICMRegistry/nls folder where the Catalog.json and widget definition file for the locale are located.
Version	Optional	String	
Categories	Optional	String	Creates a category in Case Manager Builder in which the custom page widgets in this package are listed. You can also choose to list your widgets in one of the following IBM Case Manager default categories: - Case Widgets - Utility Widgets For each category, you must provide an identifier and a title.

List of widgets in the custom widget package

Information about each widget in the package contains the following properties.

Important

Except for the definition property, these properties are identical to the properties in the widget definition file. For consistency, you can copy the values from that file into the Catalog.json file. If a value does not match, IBM Case Manager uses the value from the Catalog.json file.

Property	Required?	Type	Notes
id	Required	String	A unique identifier for the page widget.
category	Required	String	The identifier of the category in which the page widget is listed in Case Manager Builder.
title	Required	String	The name to be displayed for the page widget in Case Manager Builder.
description	Required	String	
definition	Required	String	The full path and name of the definition file for the page widget.
preview	Required	String	The relative path and name of the preview image file (.png or .gif) for the page widget.
icon	Required	String	The full path and name of the icon image file (.png or .gif) for the page widget. This image is used in the widget palette in Page Designer.
runtimeClassName	Required		The class name for the page widget as specified in the runtime plug-in for the widget package.
previewThumbnail	Required		The full path and name of the thumbnail image file (.png or .gif) for the page widget.
properties	Array		An array to define the properties that can be set for the page widget in Case Manager Builder.
events	Array	String	An array that identifies the events that the page widget publishes and subscribes to.

Page widget definition file

Introduction

You create a JSON definition file to define the following information for the custom page widget:

- Properties that define the widget
- Properties that user can set for the page widget
- Events

Page widget properties

- The properties are used to define the custom widget.

Note

The widget property values are also used for the page widget in the Catalog.json file. Refer to the table in Procedure : List of widgets in the custom widget package, page 20 for a list of properties.

- Define the properties that can be set for the widget when the user adds the widget to a page.
  - Example: Toolbars, menus, and actions that can be added to the widget.

Page widget events

Define the events that the page widget subscribes to and publishes. For each event, provide the following properties:

Property	Description
id	A unique identifier for the event.
title	The title that is displayed in the Wiring dialog box for the event.
functionName	For an event that the page widget subscribes to, the name of the function that handles the event.
type	For an event that the page widget publishes, the type of event. Set the type to Broadcast if the widget broadcasts the event to other widgets on the page. Set the type to Wiring if the event must be wired to another widget on the page.
direction	Set to either Subscribed or Published.
description	The description of the event is displayed in the hover help for the event in the Wiring dialog box.

Exercise 3.2.1: Create catalog and widget definition JSON files

Introduction

In this exercise, you develop the catalog and widget definition JSON files. You need these files to define your custom widget properties, and define any event broadcasting or event handling within your custom widget. These files are used to register the widget with the IBM Case Manager.

Procedures

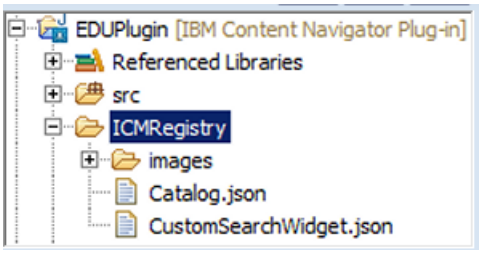
Procedure 1: Create a folder for the registry files

Sample widget definition and the catalog JSON files are included in the student system. In this procedure, you copy these files into your project and use them as a starting point.

1. Copy the folder that contains the registry files into your project.
   a. In Windows Explorer, go to the `C:\ICM\Widgets\SearchWidget` folder.
   b. Right-click the ICMRegistry folder and select Copy.

      The ICMRegistry folder contains the widget definition and the catalog JSON files. It also contains a folder with image files.
   c. In Eclipse > Package Explorer, right-click your project and select Paste to add the ICMRegistry folder.

   d. Validate that your project folder looks as in the following screen capture.



Procedure 2: Edit the catalog JSON file

Hint

You can refer to the solution file at `C:\ICM\Widgets\SearchWidget\Solution Files\Catalog_Solution.json`. Optionally, you can use the JSON Viewer application shortcut on the Windows Desktop to view and edit the JSON files.

1. In Package Explorer, expand your project > ICMRegistry folder and double-click the file `Catalog.json` to open it.

   Observe that the data fields of the custom widget package are not completed.
2. Enter the values for the fields with the data in the following table.

Section	Field	Value
General		
	Name	“EDU Custom Search Widget Package”
	Description	“IBM Case Manager custom search widget package”
	Locale	“”
	Version	“1.0”
Categories		
	id	“EDUWidgets”
	title	“EDU Widgets”
Widgets		
	id	“CustomSearchWidget”
	title	“Custom Search Widget”
	catagory	“EDUWidgets”
	description	“This widget does a custom search for cases”
	definition	“CustomSearchWidget.json”
	preview	“images/customSearch_preview.png”
	icon	“images/customSearch_icon.png”
	runtimeClassName	“eDUPluginDojo.CustomSearchWidget”
	help	“acmwrh126.htm”
	previewThumbnail	"images/customSearch_thumb.png"

3. Save your changes to the JSON file and leave the file open.

Information about the fields in the Catalog.json file:

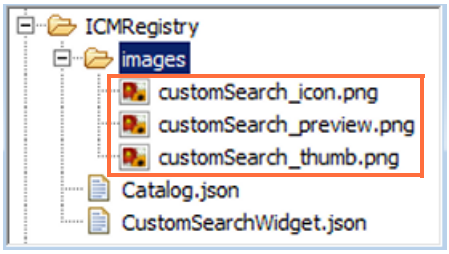
- The value of the Categories `Title` for the custom widget package is shown in the widget palette bar on the left pane in Page Designer.

```
"Categories":[
    {
        "id":"EDUWidgets",
        "title":"EDU Widgets"
    }
],
```

- ICMRegistry folder already contains the widget definition file. You specify the file name of the widget definition file (`CustomSearchWidget.json`) in the catalog.json.

```
"Widgets":[
    {
        "id":"CustomSearchWidget",
        "title":"Custom Search Widget",
        "category":"EDUWidgets",
        "description":"This widget does a custom search for cases",
        "definition":"CustomSearchWidget.json",
```

- The image files for the preview, icon, and previewThumbnail for the custom widget are already copied into the ICMRegistry folder.



- You are going to create a custom widget file with the name: `CustomSearchWidget.js`, and add to the `eDUPluginDojo` folder in the next lesson.
  - You specify the custom widget file name for the `runtimeClassName` field.

```
"icon": "images/caseinfo_icon.png",
"runtimeClassName":"eDUPluginDojo.CustomSearchWidget",
"help": "acmwrh126.htm",
```

Procedure 3: Edit the widget definition JSON file

1. In the ICMRegistry folder and double-click the file `CustomSearchWidget.json` to open it.
2. Notice that the first part of the file is similar to the `widgets` section of the Catalog.json file.
   a. Use the same values that you entered for the Catalog.json file to complete the first 10 fields of the CustomSearchWidget.json file. You can also copy and paste them.
   b. The completed file looks like the following screen capture.

```
{
        "id":"CustomSearchWidget",
        "title":"Custom Search Widget",
        "category":"EDUWidgets",
        "description":"This widget does a custom search for cases",
        "definition":"CustomSearchWidget.json",
        "preview": "images/customSearch_preview.png",
        "icon": "images/customSearch_icon.png",
        "runtimeClassName":"eDUPluginDojo.CustomSearchWidget",
        "help": "acmwrh126.htm",
        "previewThumbnail":"images/customSearch_thumb.png"
```

Procedure 4: Edit the Properties section of the CustomSearchWidget.json file

You can enter one or more properties to be shown in the settings window of your custom widget in Page Designer.

1. Set a `PreferredHeight` property for sizing of the custom widget.
   a. Set the `visibility` of this property to `false`.
      This step makes the property invisible to the user.
   b. Set the id, disabled, required, visibility, and title fields as shown in the following screen capture.

```
"properties":[
    {
        "propertyType":"property",
        "type":"string",
        "id":"PreferredHeight",
        "defaultValue":"100%",
        "disabled":true,
        "required": false,
        "visibility":false,
        "title":"Preferred Height"
    },
```

2. Define the next property as a “Custom Property 1” and set the values with the data in the following table.

Field	Value
propertyType	“property”
type	“string”
id	“customProperty1”
defaultValue	“http://”
required	false
visibility	true
style	“width:95%;”
title	“Custom Property 1”

Procedure 5: Edit the Events section of the CustomSearchWidget.json file

The events section determines whether to handle or broadcast an event. It sets the function name to handle the event, if applicable.

1. For this exercise, set the id to `icm.SearchCases` as the event id to broadcast.
2. Enter a title and description. These fields do not require specific values.
3. Because the widget is broadcasting an event to another widget, set the `direction` field to `broadcast`.

Note

if you want to handle an event that is published from another widget, you set the `direction` field to `subscribed`. You specify the same function name in this JSON file as the one that is used in your widget file to handle this event. The Subscribed event is discussed in a later lesson.

4. Your completed events section must look like the following screen capture:

```
"events":[
    {
        "id":"icm.SearchCases",
        "title":"EDU Custom Search Event",
        "functionName":"",
        "direction":"broadcast",
        "description":"This is a custom search event"
    }
]
```

5. Save the changes to the JSON file and close both the files.

Hint

You can refer to the solution file at `C:\ICM\Widgets\SearchWidget\Solution Files\CustomSearchWidget-Solution.json`.

LESSON 3.3: Implement a custom widget

What this lesson is about

This lesson describes how to implement a custom widget. For this lab exercise, you use custom search as an example. You add the widget files in your IBM Content Navigator plug-in project that you created in a previous lesson.

What you should be able to do

After completing this lesson, you should be able to:

- Implement a custom widget by creating the JavaScript files.

How you will check your progress?

- Hands on labs.

IBM Case Manager custom page widget development

Introduction

- An IBM Case Manager page widget or an IBM Case Manager custom widget is a Dojo dijit.
- In addition, the Case Manager widgets also contain the following two more items:
  - Page container events
  - Widget settings
- These two facilities enable the page widget construction and allow business analyst to dynamically adjust page behavior without any (or a little) programming skills.
- You develop the widgets in two separate steps to reuse them more easily.
  - Dojo dijit - UI component
  - Page widget wrapper

Dojo dij

- A Dojo dijit is a widget that is a part of Dojo User Interface (UI) library.
- When creating a page widget, first create a normal Dojo Dijit that is responsible for rendering the user interface and handling user interaction to collect user input.
- Design the dijit as a reusable component that takes in a context and renders itself.
- The UI dijit must not access any IBM Case Manager Page Container related APIs.
  - Accessing these APIs limits its usage to only the IBM Case Manager Case Manager Client.
  - Separation of the UI dijit from the Page widget layer allows you to reuse this dijit in a custom application or plug-in that is built on top of IBM Content Navigator.
- The dijit can be reused in a non-ECM environment if it doesn't use any Enterprise Content Manager related API.
- The dijit must contain necessary Dojo methods and events:
  - The methods and events allow the page widget wrapper or other custom widgets to exchange data or get notified by user interaction.

Page widget wrapper

After completing the UI part of the page widget, you must create a Page widget wrapper that extends this UI dijit and converts it into an IBM Case Manager page widget.

- The page widget wrapper contains the page container-related code.
  - It accepts widget settings from the page container and passes them to the underlying dijit.
  - It must also connect to Dojo events in the dijit to be notified of user actions.

- Optionally, you can separate the event handling logic from the page widget when you have a more complex event handling.
  - If you have complex logic to pre-process or post-process the event data, you can create a separate event listener object that holds the event handler implementations.
  - Then, the page widget might delegate the event handling logic to the event listener object.

Dojo Asynchronous Module Definition (AMD) loading

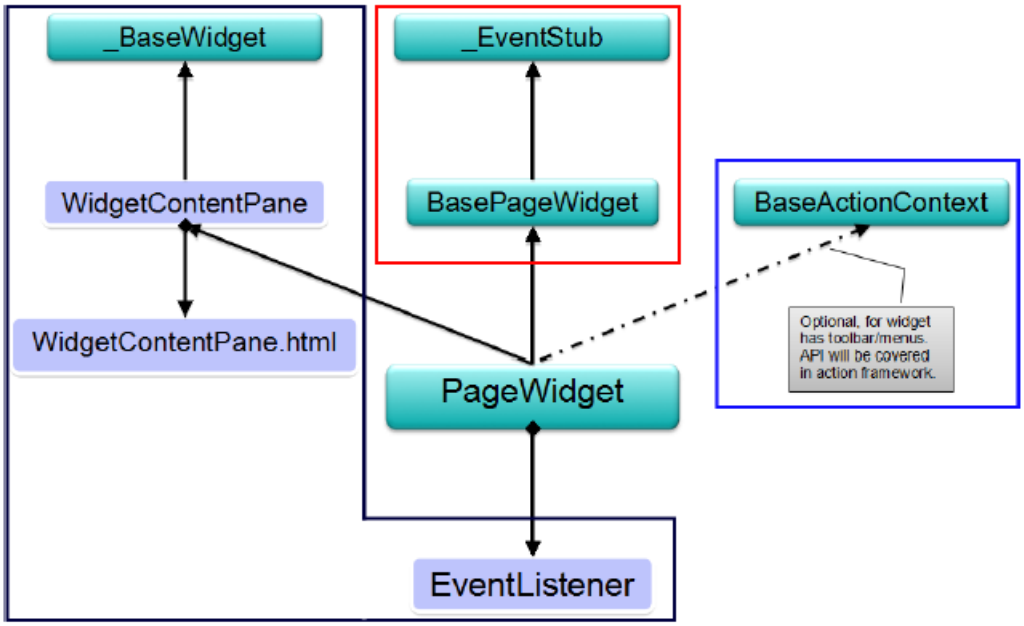
Dojo AMD loading allows the application developer to load different classes to use in the custom widget.

- The Dojo loader includes the AMD APIs.
- Use case: To define and load the IBM Case Manager or the IBM Content Navigator JavaScript API classes.

Implement a page widget

Structure for a page widget

The following graphic shows the structure for a page widget:



- In the graphic, the classes in the **black** box implement the user interface for the page widget.
  - It handles the interaction of a user with the interface.
- The classes in the **red** box are base IBM Case Manager classes from which the custom page widget inherits functions.
  - The icm.base._BaseWidget class, which provides functions that display the widget description and show or hide the content pane.
  - The icm.base._EventStub class provides functions for publishing and broadcasting methods.
- The class in the **blue** box is the base class for any page widget that hosts a toolbar or a menu.

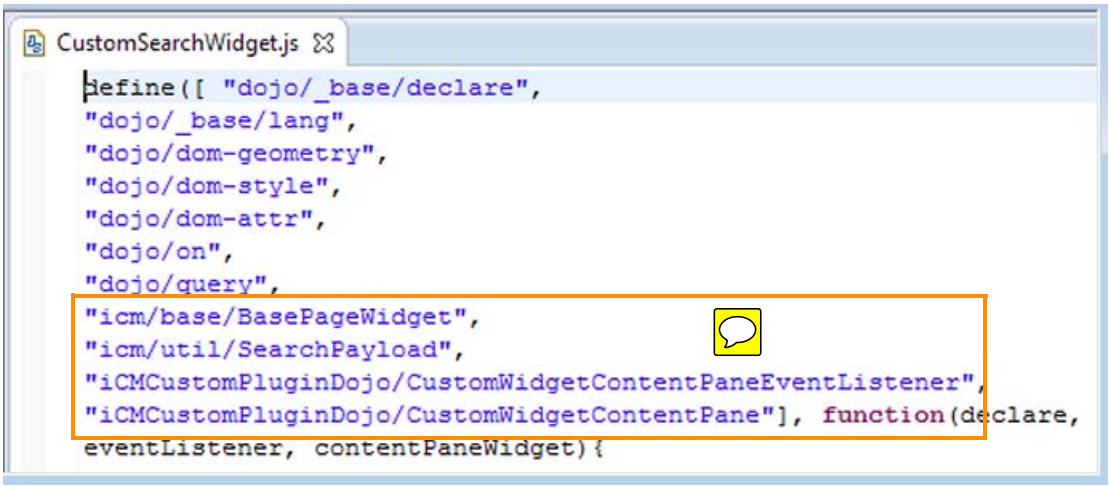
Implement a custom page widget

Following are the high-level steps to implement a custom widget.

1. Create a JavaScript file that implements the `WidgetContentPane` class.
   - The file is used to create the user interface for the CustomPageWidget widget.
   - The WidgetContentPane class Inherits from the icm.base._BaseWidget class.

- The class implements a destroy function that cleans up the user interface for the page widget when the page is closed.

2. Create a JavaScript file that implements the custom page widget.
   - The file inherits from the following classes:
     - CustomWidgetContentPane
     - BasePageWidget
     - BaseActionContext (optional, it is required for implementing toolbars and menus)



3. Write code to add a custom search function for the custom page widget.
4. If the widget subscribes an event from other widgets, add an event handler for the custom page widget.

Files that are used in this lesson

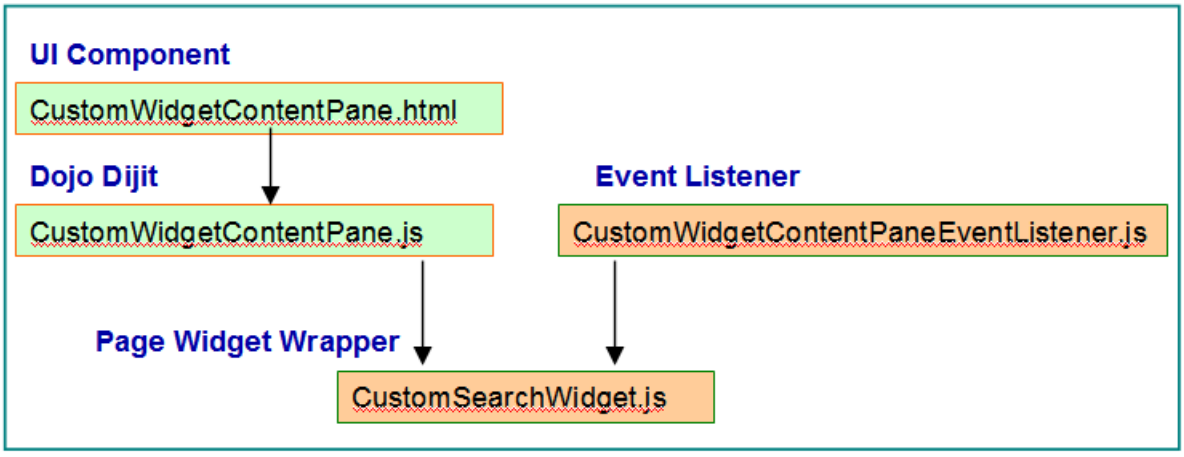
Introduction

To create a page widget, you use the following files:

- **CustomWidgetContentPane.js**
  - A Dojo dijit that implements the User Interface (UI) for your custom page widget.
  - Responsible for rendering the UI, and handling user interaction to collect user input.
  - Calls an event stub method when users click the submit button.
  - The CustomSearchWidget connects to event stub method, constructs the page widget event, and sends it out.
- **CustomWidgetContentPane.html**
  - The Dijit HTML content pane for the custom widget.
  - Add "`div`" tags or setup HTML to display when your widget renders.
- **CustomSearchWidget.js**
  - A Page widget wrapper
  - Extends the UI dijit and converts it into an IBM Case Manager page widget.
- **CustomWidgetContentPaneEventListener.js**
  - Contains the logic to pre-process or post-process the event data.
  - Separates the complex logic from the page widget wrapper class
  - Builds a custom search for cases for this lesson.

Relationship diagram

The following diagram shows how the four files are connected.



Exercise 3.3.1: Create a custom widget

Introduction

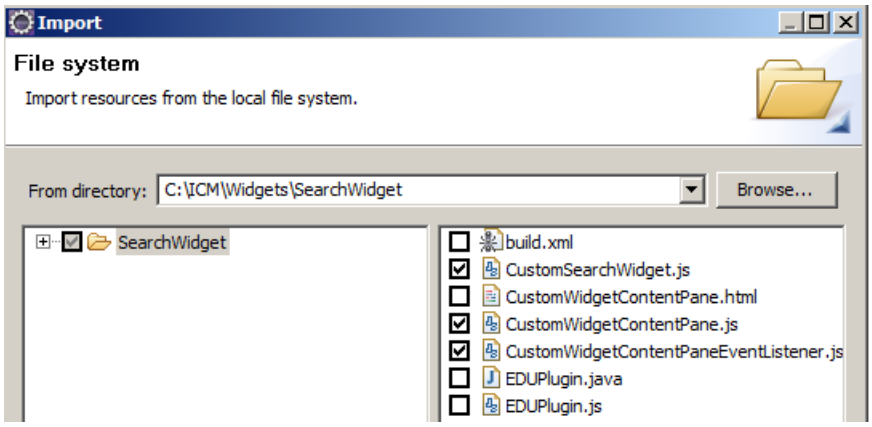
The files for this lab exercise are already created on the student image. You copy the files into your project, use them as a starting point and edit the code to complete the custom widget project. You use custom search function as an example.

Procedures

Procedure 1: Copy the files into your project

1. In Package Explorer, expand your project > `src` node and right-click `com.ibm.icm.edu.WebContent.eDUPluginDojo`
   a. Click Import from the list.
2. In the Import page, select General > File System and click Next.
   a. In the Import > File system page, click Browse.
   b. In the “Import from directory” page, go to `C:\ICM\Widgets\SearchWidget` folder and click OK.
   c. Back in the Import > File system page, select the following files:
      - `CustomSearchWidget.js`
      - `CustomWidgetContentPane.js`
      - `CustomWidgetContentPaneEventListener.js`

3. Make sure that the "Into folder" field has the following value:
   EDUPlugin/src/com/ibm/icm/edu/WebContent/eDUPluginDojo
   a. Select the "Overwrite existing resources without warning" option.



   b. Click Finish.
4. Repeat Steps 1-3 to copy the CustomWidgetContentPane.html file into the EDUPlugin/src/com/ibm/icm/edu/WebContent/eDUPluginDojo/templates package.

Hint

Notice that your project shows errors; It is because the files that you copied are not completed files. In the following procedures, you complete the code for these files.

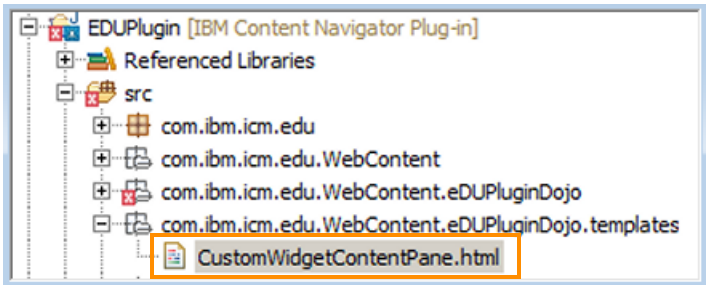
Procedure 2: Set up the Dijit HTML content pane

In this procedure, you set up the Dijit HTML content pane for your custom widget.

Note

The content pane is where you add "div" tags or setup HTML for display when your widget renders. This HTML content pane is connected to the Dijit content pane class (CustomWidgetContentPane.js) that retrieves the solution attributes and displays them on the content pane.

1. In Package Explorer, expand your project > src > com.ibm.icm.edu.WebContent.eDUPluginDojo.templates
2. Optionally, delete the wizard generated file: ConfigurationPane.html. This file is not used for this lab.
   a. Right-click the file and click Delete from the list. Click OK to confirm the delete.
3. Double-click the CustomWidgetContentPane.html file to open it.
   You dynamically populate the data in this HTML file to render your custom search widget.



4. Add a submit button to detect that the user wants to search on the user-defined values for the variables.
   a. You add this button underneath the "div" tag labeled fieldsNode.

```
<button type="button" data-dojo-attach-point="submitMeButton" class =
"submitMe">Search</button>
```

```
<form data-dojo-attach-point="searchform">
        <div data-dojo-attach-point="fieldsNode"></div>
        <button type="button" data-dojo-attach-point="submitMeButton" class = "submitMe">Search</button>
</form>
```

5. Save the CustomWidgetContentPane.html file and close it. Leave Eclipse open.

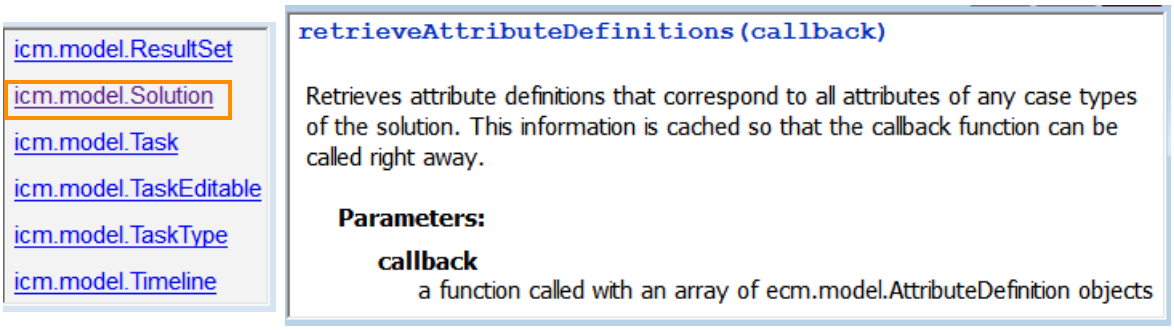
Hint

You can refer to the completed file at `C:\ICM\Widgets\SearchWidget\Solution Files\CustomWidgetContentPane-Solution.html`.

Procedure 3: Edit the CustomWidgetContentPane.js file

You load the solution attributes dynamically into the custom search widget to display the case properties for users to enter the criteria for search. In this procedure, you make a Model API call in the CustomWidgetContentPane.js file to complete the task.

- See "IBM Case Manager JavaScript API Reference" for details about the `icm.model.Solution` class.
- The `Solution` class contains the `retrieveAttributeDefinitions(callback)`function.
  - This method provides the information about the case properties of the solution.
  - The case properties are shown in the interface for the users to search upon them.



1. In Package Explorer, expand your project > src > `com.ibm.icm.edu.WebContent.eDUPluginDojo`
2. Optionally, delete the wizard generated file: `ConfigurationPane.js`. This file is not used for this lab.
   a. Right-click the file and click Delete from the list. Click OK to confirm the delete.
3. Double-click the `CustomWidgetContentPane.js` file to open it.
   a. Add the following line of code inside the `createSearchFields` function (in line 20)

```
solution.retrieveAttributeDefinitions(lang.hitch(this,this._buildSearchFields));
```

4. After the Model API call with retrieveAttributeDefintions(callback), observe that these attribute definitions are pushed into the next function in CustomWidgetContentPane.js named `_buildSearchFields(attrs)`.
   - In this lab exercise, the id and name of each attribute are used for the output.
   - The output is directly written to the content pane inner HTML.

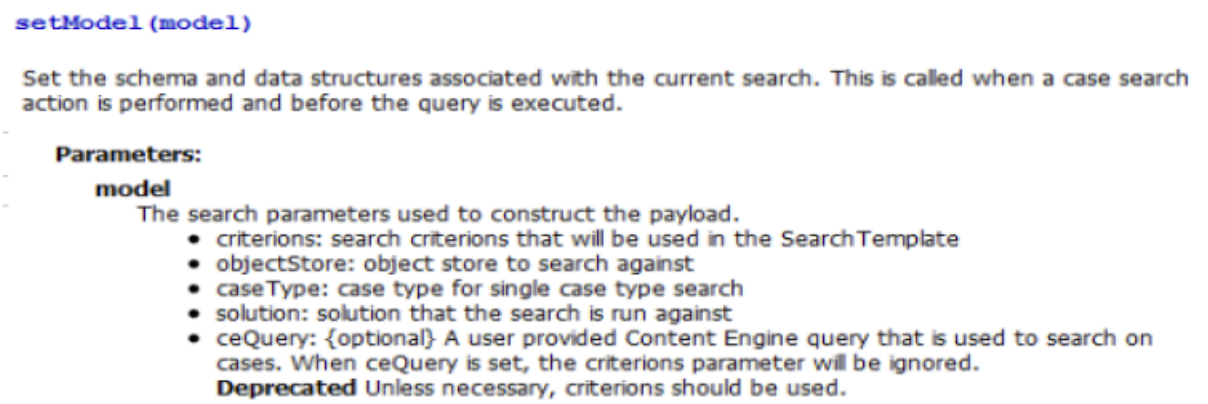
```
56      });
57      fields.push("</table>");
58      this.fieldsNode.innerHTML = fields.join("");
59
60      },
```

5. Notice that the `CustomWidgetContentPane.html` file that you edited in the previous procedure is referenced at the top of the file.
6. Observe the code in the `postCreate` function in CustomWidgetContentPane.js.
   - The user-inputted values for each attribute are sent to the CustomWidgetContentPaneEventListener to construct the searchPayload and the Model object.
   - You are going to edit the Listener in the next procedure.
7. Save and close the file.

Procedure 4: Edit CustomWidgetContentPaneEventListener.js

To run a case search, you call the `setModel(model)` function and set the schema and data structures. In this procedure, you construct the search parameters to pass them to the `setModel(model)`function.

- See “IBM Case Manager JavaScript API Reference” for details about the `icm.util.SearchPayload` class.
  - The `SearchPayload` class contains the `setModel(model)` function.



setModel(model)

Set the schema and data structures associated with the current search. This is called when a case search action is performed and before the query is executed.

Parameters:

model

The search parameters used to construct the payload.

- criterions: search criterions that will be used in the SearchTemplate
- objectStore: object store to search against
- caseType: case type for single case type search
- solution: solution that the search is run against
- ceQuery: {optional} A user provided Content Engine query that is used to search on cases. When ceQuery is set, the criterions parameter will be ignored. **Deprecated** Unless necessary, criterions should be used.

1. In Package Explorer, expand your project > src > `com.ibm.icm.edu.WebContent.eDUPluginDojo`
   a. Double-click the `CustomWidgetContentPaneEventListener.js` file to open it.
   b. Edit the file to add code in the following steps.
   c. You define the objectStore and caseType parameters for the Model object for the search payload.
2. Define the object store:
   a. Add the following code in the buildPayload: function (values) at line 44

      ```
      params.ObjectStore = solution.getTargetOS().id;
      ```
3. Define the case type for the search.
   a. Get the first case type defined in the solution.
   b. Add the following code at line 63 inside the this.widget.solution.retrieveCaseTypes(function(types)) method.

      ```
      params.caseType = types && types.length > 0 && types[0].name;
      ```

      This code defaults to the first case type.
   c. The solution is already defined after the case type at line 65.

      ```
      params.solution = solution;
      ```
4. Save and leave the file opened for the next procedure.

Procedure 5: Create the model object and broadcasting the payload

The following steps are required to accomplish search function in the custom widget:

- Set variable params with the Target Object Store, solution, case type, and ceQuer
- Pass the variable params to `setModel` function that sets the Model object.
- Call the getSearchPayload function (of the icm/util/SearchPayload class), which conducts the search and retrieves the payload to send.
- Load a variable with the payload from getSearchPayload.
- Broadcast the payload that is received to Case List widget.

Note

You set the variable params in the previous procedure.

1. Pass the variable params to the setModel function.
   a. In the CustomWidgetContentPaneEventListener.js file, add the following code at line 68 inside the this.widget.solution.retrieveCaseTypes(function(types)) method.

      `searchPayload.setModel(params);`
2. Observe the rest of the code in the file that does the following steps:
   - Calls the getSearchPayload() function and pass the payload that is received into a variable.
   - Broadcasts the payload out with the event `icm.SearchCases`.
     – The Case List widget can handle the icm.SearchCases incoming event and the payload to display the results.

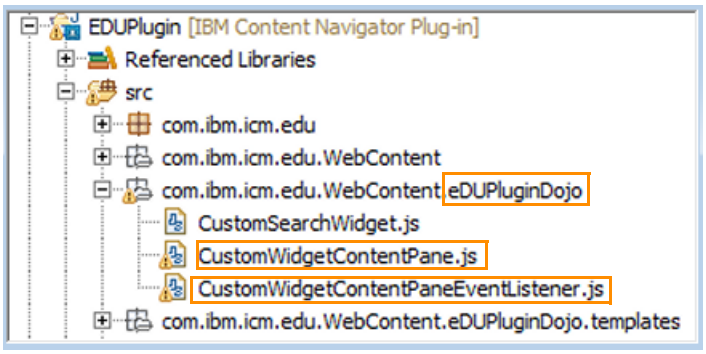
```
//getSearchPayload function and the onBroadcastEvent
var w = that.widget;
searchPayload.getSearchPayload(function(payload) {
    w.onBroadcastEvent("icm.SearchCases", payload);
    console.log(payload);
    that.displayPayload(payload);
});
```

3. Save and close the CustomWidgetContentPaneEventListner.js file.

Procedure 6: Add Dojo AMD style loading

In this procedure, you define the JavaScript files that you edited in the previous procedures with Dojo AMD loading in CustomSearchWidget.js.

1. In Package Explorer, expand your project > src > `com.ibm.icm.edu.WebContent.eDUPluginDojo`
   a. Double-click the `CustomSearchWidget.js` file to open it.
   b. Edit the file to add code in the following steps.
2. Observe that the Dojo libraries (`dojo/dom-attr` and `dojo/query`) are already loaded.
   a. The IBM Case Manager JS files (`icm/base/BasePageWidget` and `icm/util/SearchPayload`)are referenced.
3. In Package Explorer, note the path for `CustomWidgetContentPaneEventListener.js` and `CustomWidgetContentPane.js`.



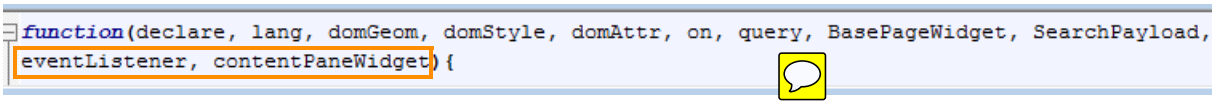
   a. Define the files in CustomSearchWidget.js. The functions in these classes become available in CustomSearchWidget.js.
   b. Complete the last two empty quotations in the define header:

   “eDUPluginDojo/CustomWidgetContentPaneEventListener.js”,

   “eDUPluginDojo/CustomWidgetContentPane.js”

```
1  define([ "dojo/_base/declare",
2  "dojo/_base/lang",
3  "dojo/dom-geometry",
4  "dojo/dom-style",
5  "dojo/dom-attr",
6  "dojo/on",
7  "dojo/query",
8  "icm/base/BasePageWidget",
9  "icm/util/SearchPayload",
10 "eDUPluginDojo/CustomWidgetContentPaneEventListene
11 "eDUPluginDojo/CustomWidgetContentPane"],
```

4. Add the two JS file definitions into the `function` method at line 12.



5. Save and close the CustomSearchWidget.js file.

Procedure 7: Edit the base JavaScript file

The base JavaScript file is loaded when IBM Content Navigator loads the plug-in. You use this JavaScript file to load any JavaScript classes that must be available throughout the session.

1. In Package Explorer, expand your project > src > `com.ibm.icm.edu.WebContent`.
   a. Double-click the `EDUPlugin.js` file to open it.
   b. Edit the file to add code in the following steps.
2. Observe that the wizard generated this file with sample code.
   a. Edit the file to load CustomSearchWidget.js as shown in the following code:

```
require(["eDUPluginDojo/CustomSearchWidget"],
function(CustomSearchWidget) {
});
```

3. Save and close the EDUPlugin.js file.

Note

You package, register, and test your widget package in the next lesson.

LESSON 3.4: Build and register a widget package

What this lesson is about

This lesson provides guidance for building and registering the widget package. You create a compressed file that contains the Registry folder and the plug-in JAR file, register with IBM Content Manager and test your widget.

What you should be able to do

After completing this lesson, you should be able to:

- Build and register the widget package

How you will check your progress?

- Hands on labs.

Register the custom widget package

Contents in the custom page widget package

A Case Manager custom widget compressed file (ZIP) must include the following items:

- IBM Content Navigator plug-in JAR file
  - If you have a small widget customization, you can add the page widget code as part of the plug-in JAR file. No need for a separate EAR file.
- Optional EAR file
  - If you have a large custom application, then you package the custom widget code as an EAR file separately.
- Custom widget definition JSON file (in a folder)
- Custom widget catalog JSON file (in a folder)

Options for widget deployment and registration

Use one of the following two options for deploying and registering your custom widget package. The option depends on how you packaged the widgets:

- Option 1 - If you packaged your widgets as a JAR file:
  - Use the IBM Case Manager administration web client to register your custom widget.

Note

Lessons 1- 4 in this unit provides details for the JAR file approach.

- Option 2 - If you packaged your widgets as an EAR file:
  - Use the IBM Case Manager configuration tool to register your custom widget.

Note

Lessons 5 and 6 in this unit provides details for the EAR file approach.

Exercise 3.4.1: Build and register the widget package

Introduction

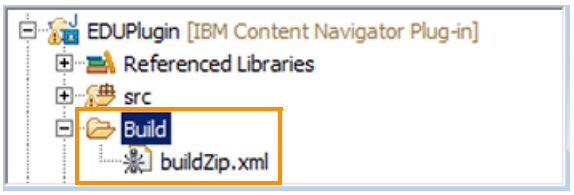
- The wizard-generated `build.xml` file packages the Content Navigator plug-in files into a JAR file.
- To register a Case Manager widget, you require a compressed file (ZIP) that includes the plug-in JAR file and the ICMRegistry folder that contains the catalog and widget definition files.
- In this exercise, you create the ZIP file.

Procedures

Procedure 1: Package the widget files into a ZIP file

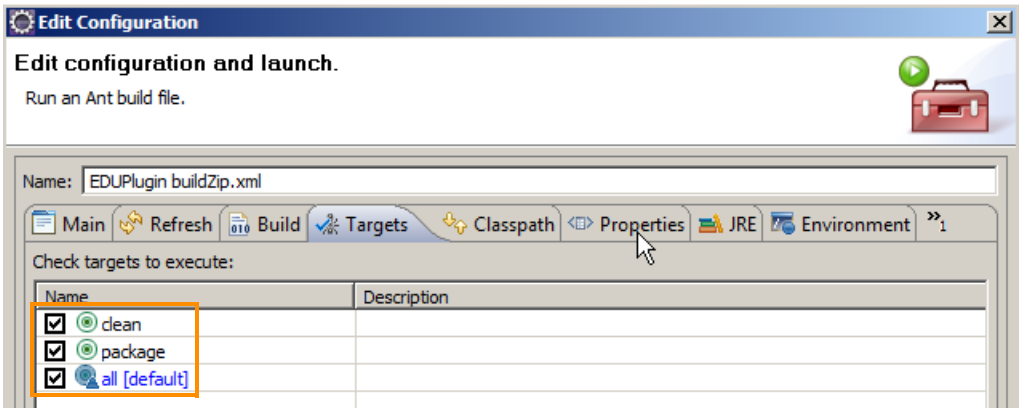
The `buildZIP.xml` file that packages your project into a ZIP file is included in the student system. In this procedure, you copy the file into your project and generate the package.

1. In Package Explorer, expand your project.
   a. Optionally, open the `build.xml` file to observe the contents of the file.
2. Copy the folder that contains the `buildZIP.xml` file into your project.
   a. In Windows Explorer, go to the `C:\ICM\Widgets\SearchWidget` folder.
   b. Right-click the `Build` folder and select Copy.
   c. In Eclipse > Package Explorer, right-click your project and select Paste.
   d. Validate that your project folder looks as in the following screen capture.

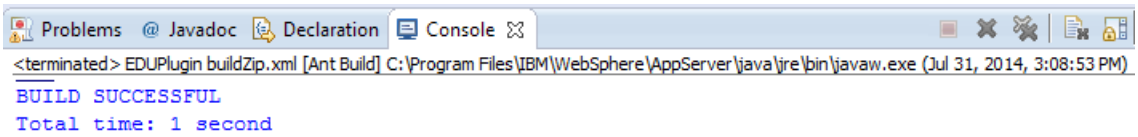


3. In Package Explorer, expand your project > Build.
   a. Open the `buildZIP.xml` file and observe the contents of the file.
      This file calls the `build.xml` that creates the JAR file; then, it packages the JAR file and the ICMRegistry folder into a ZIP file.
   b. Right-click the `buildZIP.xml` and select Run As > “2.Ant Build” from the list.

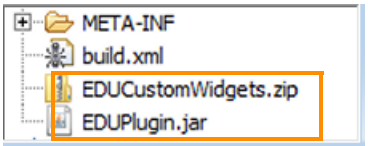
4. In the “Edit Configuration” window, select all the targets under the Name column and click Run at the end of the page.



5. Validate that the console tab at the bottom pane displays that the Build was successful.
   a. Verify that the JAR, and ZIP files are created.
   b. Ignore the “includeantruntime” warning.



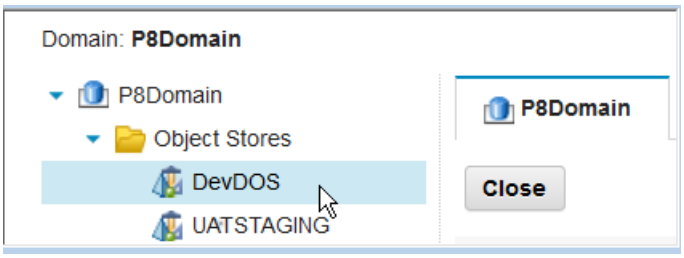
6. Right-click your project and select Refresh from the list.
   a. Verify that the JAR and ZIP files are listed in your project.



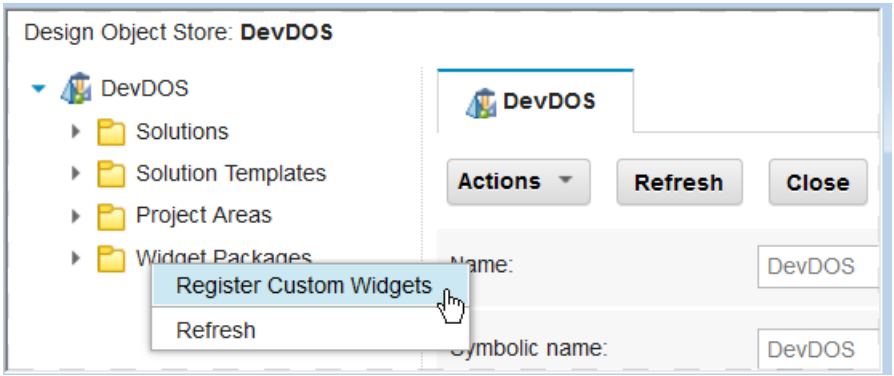
Procedure 2: Register the custom widget

In this procedure, you register your custom widget package in the IBM Case Manager admin tool.

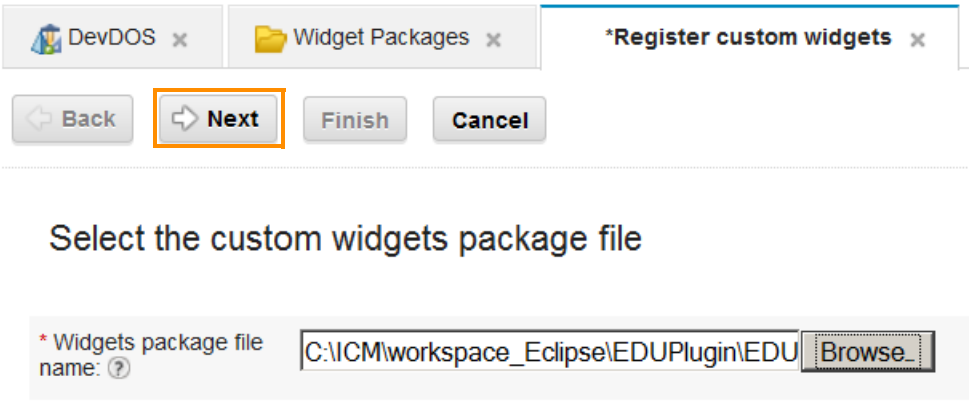
1. Start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Open the Design Object Store.
   a. Click P8Domain > Object Stores > DevDOS in the left pane.
   The DevDOS tab opens.



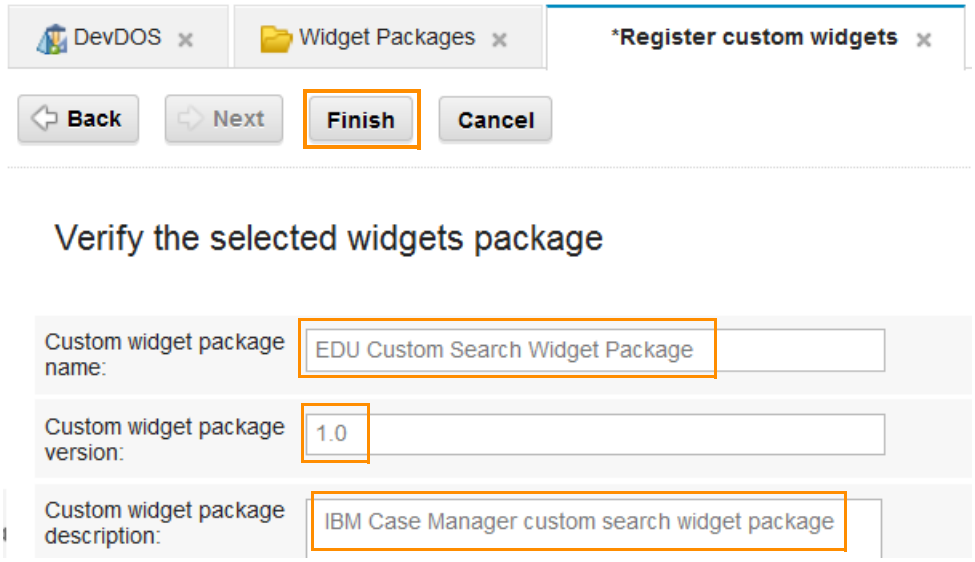
3. In the DevDOS tab > left pane, expand the DevDOS.
   a. Right-click “Widget Packages” and select “Register Custom Widgets” from the list.
   The “Register custom widgets” tab opens.



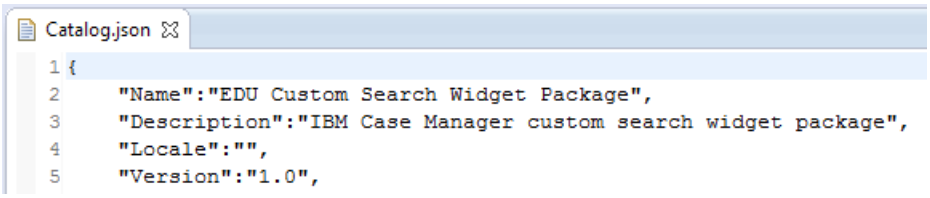
4. In the “Register custom widgets” tab, click Browse.
   a. Go to the folder where the ZIP file for your widget package is created: `C:\ICM\workspace_Eclipse\EDUPlugin`
   b. Select `ICMCustomWidgets.ZIP` and click Open.
   c. Verify that the path is added to the “Widget package file name” field and click Next.



5. In the “Verify the selected widgets package” page, check the widget package name, version, and the description.

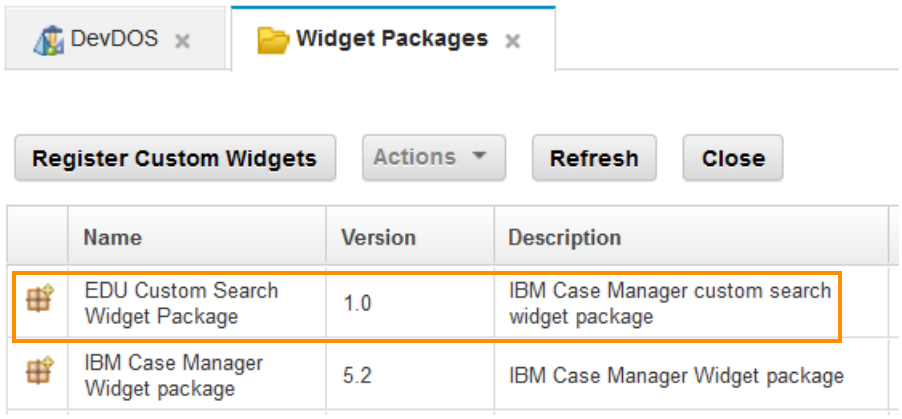


a. Recall that the same values that you added to your Catalog.json file as shown in the following screen capture (Lesson 2 in this unit).

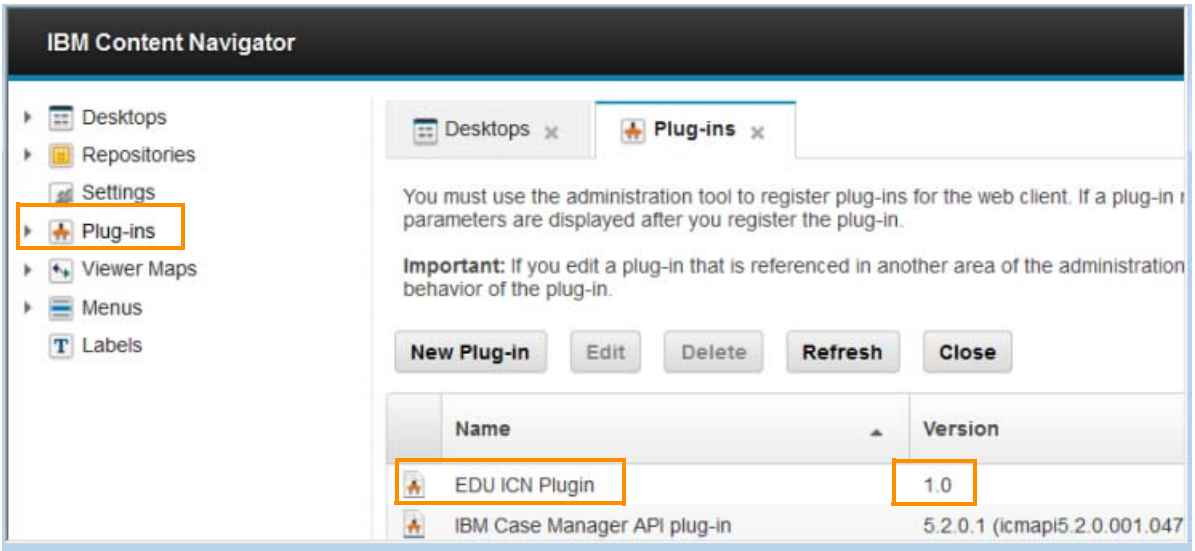


b. Click Finish.

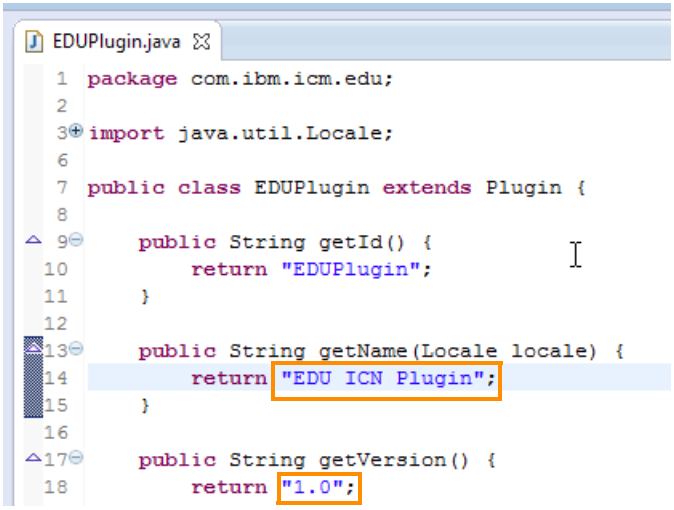
6. When you get the “The widget package was successfully registered”, click Close.
7. Check that your widget package is listed in the “Widget Packages” tab.



8. Verify that the custom plug-in is registered:
   a. Start the IBM Content Navigator Admin desktop.
      – URL: `http://ecmedu01:9080/navigator/?desktop=admin`
      – User name: `P8admin`
      – Password: `IBMFileNetP8`
   b. In the IBM Content Navigator Admin desktop, select Plug-ins in the left pane.
      The Plug-ins tab opens.
   c. In the Plug-ins tab, verify that your plug-in is listed.



   d. Recall that the same values that you added to your EDUPlugin.java file as shown in the following screen capture (Lesson 12 in this unit).



9. Log out of the admin tools and close the browser.

Exercise 3.4.2: Test the custom widget

Introduction

In this exercise, you create a custom Solution Page in Case Manager Builder, add your custom widget to the page, and test it.

Procedures

Procedure 1: Create a custom page

A solution for this lab is already created. In this procedure, you copy the default Cases page and modify it.

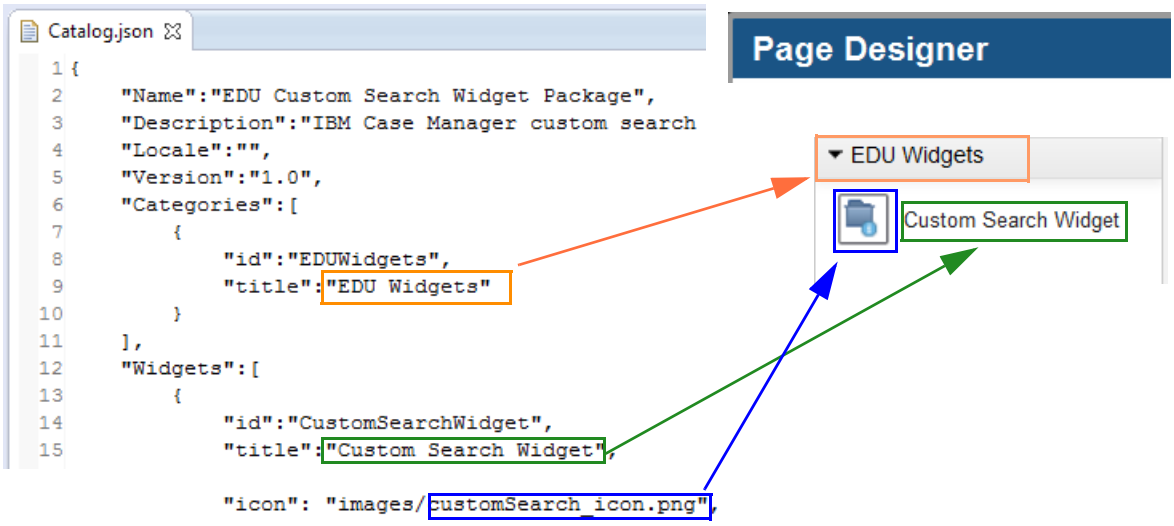
1. In Firefox, log on to IBM Case Manager Builder as an administrative user.
   - URL: `http://ecmedu01:9080/CaseBuilder`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.

   Hover the mouse over the solution to see the links.
3. Create a custom page:
   a. Open the Pages tab and expand the Solution Pages.
   b. Hover the mouse over the Cases page name.
   c. Select the Copy icon on the right side of the page.
   d. In the resulting page, edit the name to `Custom Search` for your new page and click OK to create the copy.
   e. Save your work by clicking Save at the top of the page.
4. Leave the Pages tab opened for the next procedure.

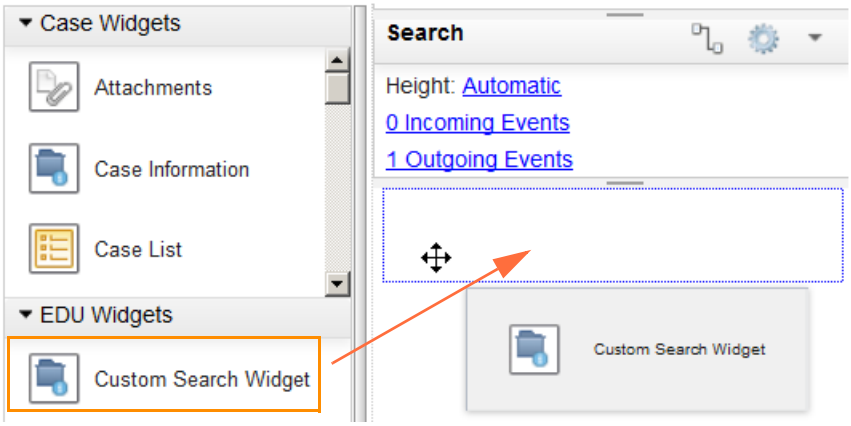
Procedure 2: Edit the page to add the custom widget

In this procedure, you edit the page to add your custom search widget.

1. In the Pages tab, double- click your page (`Custom Search`)to edit it in Page Designer.
2. Observe that the title of the widget category (“EDU Widgets”) is displayed in the left pane of the Page Designer.
   a. The title of the widget that you added is displayed under the widget category in which it is defined. The widget is available to add to the pages.
   b. Recall the values that entered in the catalog.json file in your widget package.



3. Drag the “Custom Search Widget” (under the “Custom Widgets” section) from widget palette on the left column to the page on the right.
   a. Place the widget below the existing default Search widget.



4. Click Save to save your work.
   a. Click Save and then Close to close Page Designer.
   b. Leave the solution open for the next procedure.

Procedure 3: Assign the custom page to a role

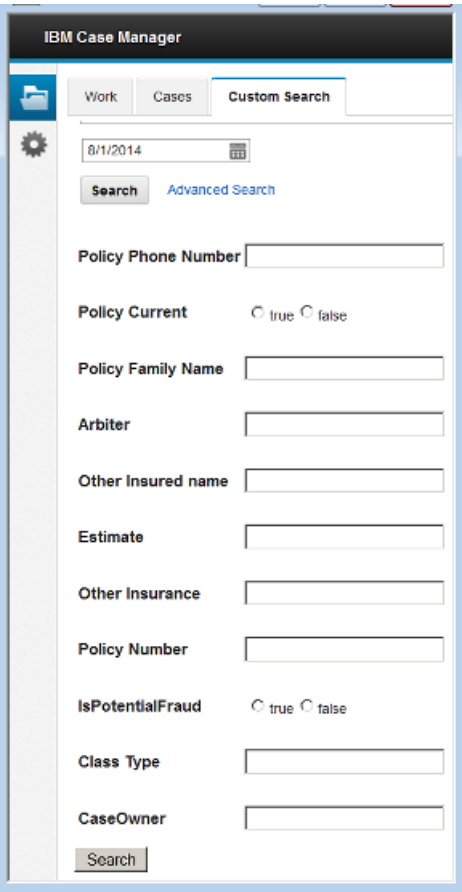
1. Your solution is already opened in the Case Manager Builder. Open the Roles tab.
2. Click the Customer Service Rep role link.
   a. Open the Pages subtab.
3. Assign the new page that you created.
   a. Click Assign Page.
   b. Select the Custom Search Page.
   c. Click OK to close the dialog window.
4. Verify that your page is listed in the Pages tab.
   a. Click OK.
   b. Click OK All to accept the changes to the role.
   c. Click “Save and Close” at the top of the page to exit the solution editor.
   d. Leave the Case Manager Builder open for the next procedure.

Procedure 4: Redeploy the solution

1. In the Manage Solutions page, select Lab Claims Solution and hover the mouse over.
   a. Click Commit to save the changes to the repository.
   b. Click “Commit My Changes” in the Confirmation window.
2. Select the solution again, hover the mouse over, and click Deploy.
   c. Wait for the green check mark to appear next to the solution.
3. Hover the mouse over the solution and click Test to open Case Manager Client.

Procedure 5: Test the custom search widget

1. In the Case Manager Client, select the Custom Search tab to open your custom page.
2. Verify that the custom Search fields are shown.



3. Search for cases.
   a. In the Custom Search widget section, enter `%` in the text box for the Policy Family Name field and click the Search button.
   b. Verify that available cases are listed in the Case List widget.
4. Optionally, enter a specific Policy Family Name (Example: `Smith`) and verify that it returns only one case that case the Value `Smith.`

Exercise 3.4.3: Troubleshooting (if the widget does not work)

Introduction

Troubleshooting

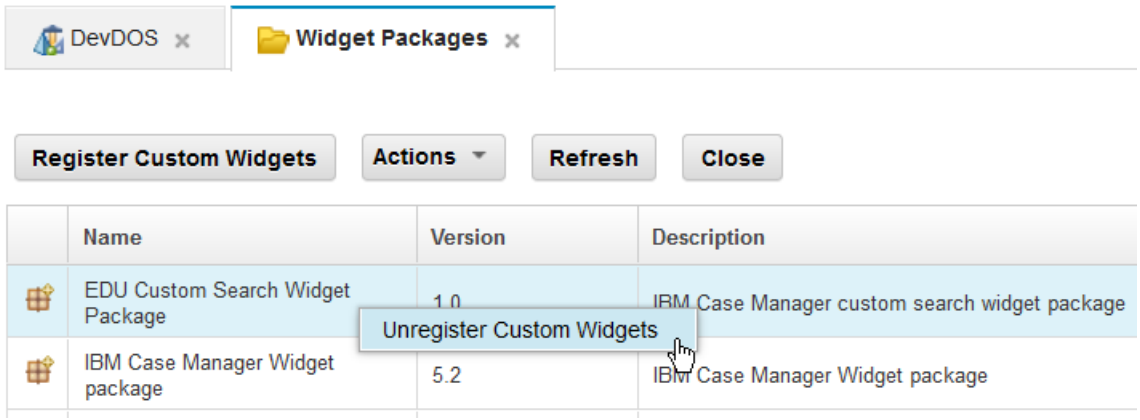
If you get errors when you test your custom widget in the Case Manager Client, do the following steps.

1. Check your code and if needed, replace your code with the solution files.
2. Package the code again into a ZIP file.
   a. Refer to Procedure 1: Package the widget files into a ZIP file, page 44 for more details.
3. Unregister the existing widget package.
   a. Refer to Procedure 1: Unregister the custom widget, page 54 for more details.
4. Repeat the registration with the new package.
   a. Refer to Procedure 2: Register the custom widget, page 46 for more details.
5. Verify the Content Navigator plug-in.
   a. Refer to Procedure 2: Verify the Content Navigator plug-in, page 55 for more details.
6. Clear browser cache.
   a. The student image is configured to clear the cache when you close the browser.
7. Test your widget.
   a. Refer to Procedure 5: Test the custom search widget, page 53 for more details.

Procedure 1: Unregister the custom widget

In this procedure, you unregister your custom widget package in the IBM Case Manager admin tool.

1. Start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Open the Design Object Store.
   a. Click P8Domain > Object Stores > DevDOS in the left pane.
      The DevDOS tab opens.
3. In the DevDOS tab, expand the DevDOS and select “Widget Packages” in the left pane.
   a. In the Widget Packages tab, right-click your widget package and select “Unregister Custom Widgets”.
      The “Unregister Custom Widgets” tab opens.



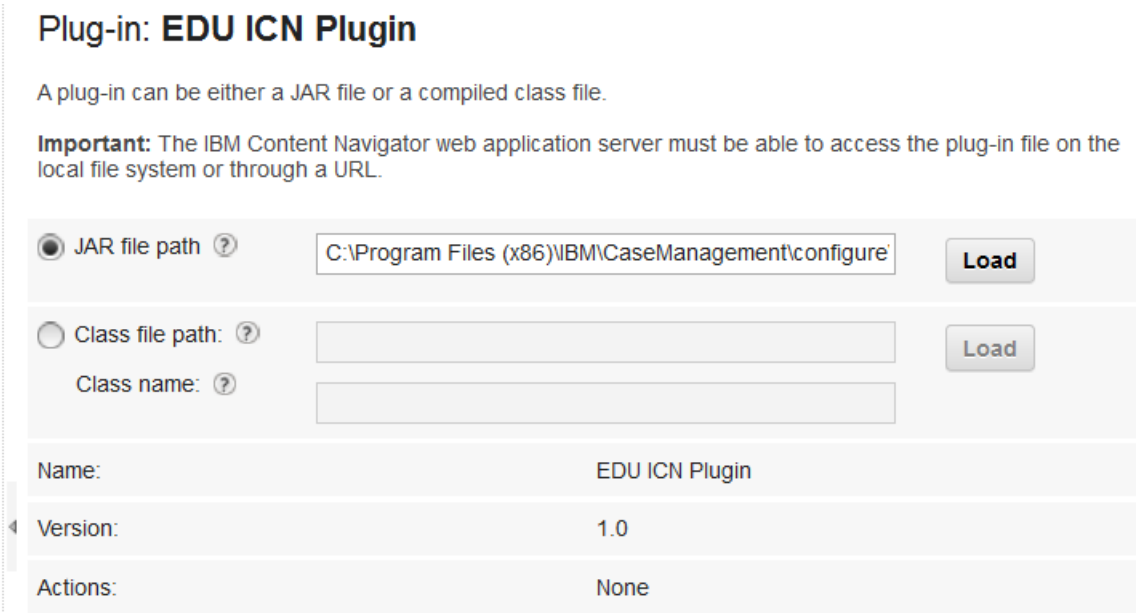
4. Click Finish.
   a. When you get the message that the package is removed, click Close.
   b. Verify that your widget package is removed from the list in the Widgets Packages tab.
5. Log out of the IBM Case Manager admin tool and close the browser.

Procedure 2: Verify the Content Navigator plug-in

In this procedure, you verify that the plug-in that you developed is registered properly in the IBM Content Navigator admin tool.

1. Start the IBM Content Navigator Administration Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=admin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Open the Plug-ins tab.
   a. Select Plug-ins in the left pane.
3. In the Plug-ins tab, select your plug-in (`EDU ICN Plugin`) and click Edit.

4. In the `EDU ICN Plugin` tab, click Load for the JAR file path.
   a. Verify that the details about the plug-in are displayed as shown in the following screen capture.



5. Click Save and Close.
6. Log out of the IBM Content Navigator Administration Desktop and close the browser.

LESSON 3.5: Create a widget with a toolbar and a menu

What this lesson is about

This lesson describes how to create a widget with toolbar and menu actions. You also define widget properties, and add event handling to your widget. In the lessons 1-4, you created an IBM Content Navigator plug-in project, added widgets as part of the plug-in, and registered the widget in IBM Case Manager administration client.

In this lesson, you create a web module for your widgets along with the plug-in, and deploy the widget in IBM Case Manager configuration tool.

What you should be able to do

After completing this lesson, you should be able to:

- Create a widget with a toolbar and a menu.
- Define widget properties and add event handling to your widget.
- Create a web module for the widgets.

How you will check your progress?

- Hands on labs.

Folder structure for the custom widget project in this lesson

Sub-projects and folders for the primary project

In this lesson, you use the following sub-projects and folders for the primary project that is required for packaging your custom widgets:

- **ICMCustomPlugin**
  - This project contains the files that are required to create a plug-in for IBM Content Navigator.
  - You can name this project as wanted.
  - You create a JAR file from this project.
- **ICMCustomWidget**
  - This project is used for the web module.
  - It contains the package structure and the files for the custom widget.
  - You can name this project as wanted.
  - You create an EAR file from this project.
- **ICMRegistry**
  - This folder provides the widget definition and catalog JSON files for your widget.
  - This folder name is required.
- **Build**
  - This folder contains the build XML files to package your project into a ZIP file.

Implement toolbar and menu for your widget

Menu implementation

IBM Case Manager JavaScript API provides the following class to include menu items in your widget: `icm.widget.menu.ContextualMenu`

This class

- Extends icm.widget.menu.Menu.
- Represents a contextual menu.
- Defined in: <icm/widget/menu/ContextualMenu.js>.

Toolbar implementation

IBM Case Manager JavaScript API provides the following class to include toolbar items in your widget: `icm.widget.menu.Toolbar`

This class

- Extends icm.widget.menu.Menu.
- Represents a toolbar.
- Defined in: <icm/widget/menu/Toolbar.js>.

Exercise 3.5.1: Create a Java project in Eclipse

Introduction

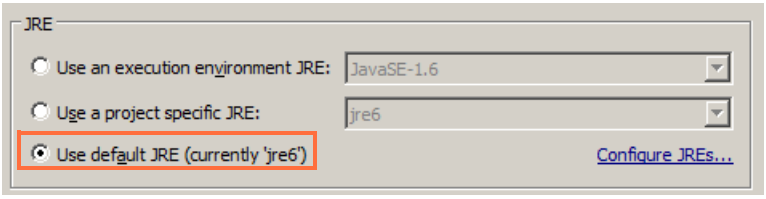
In this exercise, you create a Java project, copy the folders with files for the project, and review the files.

Procedures

Procedure 1: Create a Java project in Eclipse

1. Open Eclipse by double-clicking the Eclipse icon in your desktop.
   a. In the Workspace Launcher page, leave the default workspace directory (`C:\ICM\workspace_Eclipse`) and click OK.
2. Open the project creation wizard.
   a. In Eclipse, click File > New > Project > Java Project.

      The New Java Project page opens.
3. Create a Project.
   a. In the `New Java Project` page, enter a Project Name (Example: `CPageW` - for “Custom Page Widget”) and click Next.
   b. In the JRE section, select the “Use default JRE” option. Make sure `jre6` is selected.



   c. Leave other default settings and click Next.
   d. Click Finish.
   e. Click Yes, if you are prompted to open the Java Perspective.

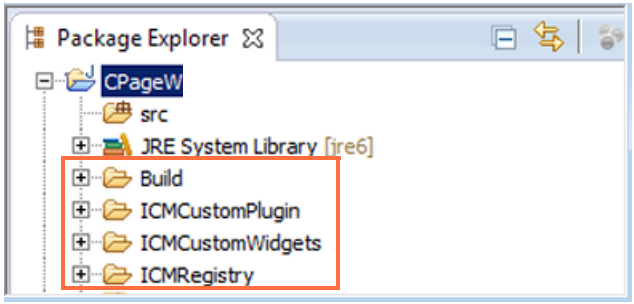
Procedure 2: Copy the folders for your project

The files that are required for this project are included in the student system. In this procedure, you copy these files into your project and use them as a starting point.

For this lab exercise, you use the following four folders:

1. In Windows Explorer, go to the `C:\ICM\Widgets\CustomPageWidget` folder.
   a. Right-click the `ICMCustomPlugin` folder and select Copy.
   b. In Eclipse > Package Explorer, right-click your project and select Paste.
2. Repeat Step 1 to copy the following folders into your project.
   - `ICMCustomWidgets`
   - `ICMRegistry`
   - `Build`

   a. The directory must look like the following screen capture:



Procedure 3: Review the IBM Content Navigator custom plug-in

The IBM Content Navigator custom plug-in for this project is similar to the one that you did in Lesson 1. In this procedure, you check the code that is provided for adding actions.

1. In Eclipse > Package Explorer, expand your project > ICMCustomPlugin > src > com > ibm > icm > extension > custom.
   a. Open the `ICMCustomPlugin.java` file.
2. The method `public String getId` returns “ICMCustomPlugin” in line 18. This value provides an identifier for the plug-in.

```
16      public String getId() {
17          // TODO Auto-generated method stub
18          return "ICMCustomPlugin";
19      }
```

3. Observe the method public `PluginAction[] getActions()` near line 35. It provides a list of actions that this plug-in adds to the main toolbar of the web client.

```
public PluginAction[] getActions() {
    return new PluginAction[] { new CustomAddCaseAction(), new
    CustomAddToAttachmentAction(), new CustomAddTaskAction(), new
    CustomAddCasePerRoleAction()};
}
```

4. Close the file without any changes.

Note

The actions are defined under the ICMCustomPlugin > src > com > ibm > icm > extension > custom > actions folder. For more information, see the “F120: IBM Content Navigator 2.0.2: Customize and Extend the Features” course, and the ‘Customizing and Extending IBM Content Navigator” RedBook.

5. In Eclipse > Package Explorer, expand your project > ICMCustomPlugin > src > com > ibm > icm > extension > custom> WebContent.
   a. Open the `ICMCustomPlugin.js` file that is the base JavaScript file.

      It is loaded when IBM Content Navigator loads the plug-in. You can use this JavaScript file to apply any global changes (such as a style override) or load any JavaScript classes that must be available throughout the session.
   b. Observe that the context root for the web module of the custom widget package is specified.
   c. The code that is shown in the screen capture registers the Dojo module path `icm/custom` for the custom runtime code.

```
3  var icmContextRoot = "/ICMCustomWidgets";
```

```
45 //setup ICM runtime
46 dojo.setObject("ecmwdgt.contextRoot", icmContextRoot);
47
48 var paths = {
49         "icm/custom":"/ICMCustomWidgets/icm/custom",
50         "icm":"/ICMClient/icm"
51     };
52 require({paths:paths});
```

   d. In the following lines of this file, CSS and other scripts are loaded.
6. Close the file without any changes.

Exercise 3.5.2: Create the widget definition JSON file

Introduction

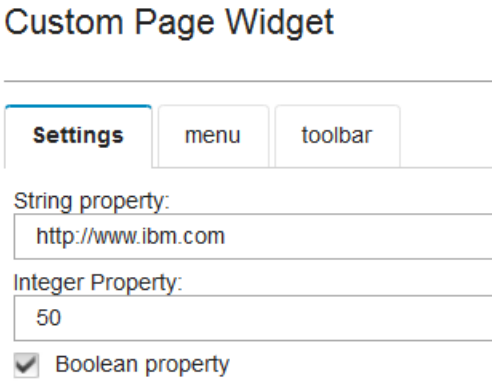
You already copied a folder that contains the widget definition and the catalog JSON files into your project. In this procedure, you review the contents in the files.

Procedures

Procedure 1: Define widget properties (attributes)

The widget definition and the catalog JSON files that are used in this project are similar to the one that you created in Lesson 2.

1. In Package Explorer, expand your project > ICMRegistry folder and double-click the file `Catalog.json` to open it.
   a. Review the values for the fields.
2. In the ICMRegistry folder, double-click the widget definition JSON file, CustomPageWidget.json to open it.
3. Observe the "`properties`" section > "`propertyType`":"`property`" blocks.
   – You can enter one or more widget properties.
   – For this exercise, String, Integer, and Boolean properties are used as an example.
   – The properties are displayed in the settings tab of your custom widget configuration page.
   – In Case Manager Builder, the users can access the configuration page from the Page Designer, and assign values during the design time.



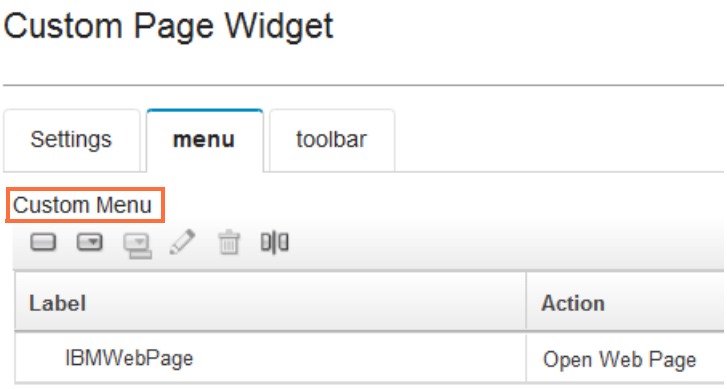
Procedure 2: Define the menu action for the custom widget

The menus allow the user to right-click the custom widget to get a list of options to invoke.

1. Observe the "properties" section > "propertyType":"group" > "id": "Menu" block.
   a. Verify that the "type" field has "contextualMenu" as the value.
2. The "id" value ("customContextualMenu") and the "context" value ("CustomContext") that are specified in this file must be used in the widget JavaScript file.
3. The menu has an actionList that might be an array of values. For this lab, "Open Web page" is used an example.

```
"propertyType":"group",
"type":"tab",
"id":"Menu",
"title":"menu",
"propertyMembers":[
    {
        "propertyType":"property",
        "type":"contextualMenu",
        "id":"customContextualMenu",
        "context": ["CustomContext"],
        "defaultValue":{
            "actionList": [
                {
                    "actionDefinitionId": "icm.action.utility.OpenWebPage",
                    "propertiesValue": {
                        "label": "IBMWebPage",
                        "targetURL": "http://www.ibm.com"
                    }
                }
            ]
        },
        "required":false,
        "visibility":true,
        "title": "Custom Menu"
```

4. The Custom Menu that is defined in this section is displayed in the menu tab of the custom widget configuration page as shown in the following screen capture.
   a. The value for the "title" ("Custom Menu"), the "label" ("IBMWebpage") and the action name are also displayed on the page.
   b. In Case Manager Builder, the users can access the page from the Page Designer during the design time to assign values.



Procedure 3: Define the toolbar action for the custom widget.

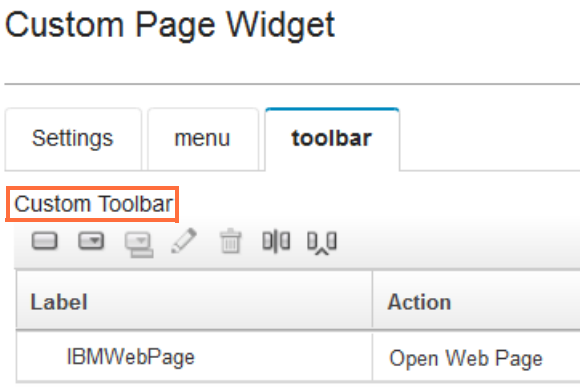
A toolbar with a button allows the user to click it to start an action.

1. Observe the "properties" section > "propertyType":"group" > "id": "Toolbars" block.

```
"propertyType":"group",
"type":"tab",
"id":"Toolbars",
"title":"toolbar",
"propertyMembers":[
    {
        "propertyType":"property",
        "type":"toolbar",
        "id":"customToolbar",
        "context": ["CustomContext"],
        "defaultValue":{
            "actionList": [
                {
                    "actionDefinitionId": "icm.action.utility.OpenWebPage",
                    "propertiesValue": {
                        "label": "IBMWebPage",
                        "targetURL": "http://www.ibm.com"
                    }
                }
            ]
        },
        "required":false,
        "visibility":true,
        "title": "Custom Toolbar"
```

2. Verify that the "type" field has "toolbar" as the value.

3. The “id” value (“customtToolbar”) and the “context” value (“CustomContext”) that are specified in this file must be used in the widget JavaScript file.
4. The toolbar has an actionList that might be an array of values. For this lab, “Open Web page” is used an example.
5. The Custom toolbar that is defined in this section is displayed in the toolbar tab of the custom widget configuration page as shown in the following screen capture.



a. The value for the “title” (“Custom Toolbar”), the “label” (“IBMwebpage”) and the action name are also displayed on the page.
b. In Case Manager Builder, the users can access the page from the Page Designer during the design time to assign values.

Procedure 4: Define the events for the custom widget

This section determines whether to handle or broadcast an event and sets the function name to handle the event if applicable.

1. Verify that the id is set to icm.CustomEvent.

   This event id allows the custom widget to handle any event that is wired to the widget.

```
"events":[
    {
        "id":"icm.CustomEvent",
        "title":"Custom Event 1",
        "functionName":"handleICM_CustomEvent",
        "direction":"subscribed",
        "description":"Custom Event 1"
    }
]
```

2. The values for title and description of the event are not specific like icm.CustomEvent. You can provide any string for the title and description fields.
3. Set the `functionName` and `direction` fields for the event.
   a. Because the widget is handling an event from another widget, verify that the direction is set to "`subscribed`".

Note

Recall that in lesson 2, you set the direction to broadcast, because the custom search widget sends an event to another widget.

   b. The functionName is `handleICM_CustomEvent`.

      You use this function name in the widget JavaScript file. The names in both files must match.
4. Close the widget definition file without any changes.

Exercise 3.5.3: Implement your widget

Introduction

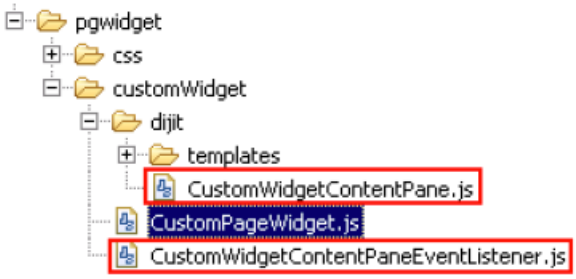
In the previous exercise, you copied the folder that contains the widget implementation files to your project. In this exercise, you review the code and edit the files.

Procedures

Procedure 1: Set up Dojo AMD loading

In this procedure, you add Dojo AMD loading into CustomPageWidget.js.

1. In Eclipse > Package Explorer > your project, expand `ICMCustomWidgets > WebContent > icm > custom > pgwidget > customWidget`
2. Open the CustomPageWidget.js file.
   - Notice that the Dojo libraries are already loaded (Example: `dojo/dom-style`).
   - Some of the IBM Case Manager Java Script files are also referenced (Example: `icm/base/BasePageWidget`).
3. Define your Java Script files in the custom widget package with Dojo AMD.
   a. In your Eclipse project navigation pane on the left, note the path for `CustomWidgetContentPaneEventListener.js` and `CustomWidgetContentPane.js`.



   b. You define these files in the CustomPageWidget.js file so that you can call the functions in those classes.

c. Complete the last two empty quotation marks in CustomPageWidget.js file in the define header as shown in the screen capture.

```
define([ "dojo/_base/declare",
    "dojo/_base/lang",
    "dojo/dom-geometry",
    "dojo/dom-style",
    "icm/base/BasePageWidget",
    "icm/custom/pgwidget/customWidget/CustomWidgetContentPaneEventListener",
    "icm/custom/pgwidget/customWidget/dijit/CustomWidgetContentPane",
```

d. Add the two JS file definitions into the `function` method after the define header as shown in the screen capture.

```
"icm/base/BaseActionContext"], function(declare, lang, domGeom, domStyle, I
    eventListener, contentPaneWidget, MenuManager, toolBar, ContextualMenu,
```

4. Define the following IBM Case Manager JavaScript API classes to make the menu and toolbar features available in the custom widget.

   `"icm/widget/menu/ContextualMenu"`

   `"icm/widget/menu/Toolbar"`

   a. Add the paths for these two classes into the define section and into the function section.

   b. The completed code looks like the one in the screen capture:

```
"icm/widget/menu/MenuManager",
"icm/widget/menu/Toolbar",
"icm/widget/menu/ContextualMenu",
"icm/base/BaseActionContext"], function(declare, lang, domGeom, domStyle, I
    eventListener, contentPaneWidget, MenuManager, toolBar, ContextualMenu,
```

5. Save the changes and close the file.

Hint

You can refer to the completed file at `C:\ICM\Widgets\CustomPageWidget\Solution Files\CustomPageWidget-Solution.js`.

Procedure 2: Expose widget properties (attributes) in the content pane

In the previous lab exercise, you defined custom properties for your widget in the widget definition file.

- The properties are displayed in the settings tab of your custom widget configuration page.
- In Case Manager Client, the users can access the configuration page from the Page Designer, and assign values during the design time.
- To see the values for these properties in the custom widget at the run time, you must expose the properties on the widget's content pane.

In this procedure, you edit the HTML file for the custom widget content pane, and then set up in the CustomWidgetContentPaneEventListener.js file to show the properties on the content pane.

1. In Eclipse > Package Explorer > your project, expand `ICMCustomWidgets > WebContent > icm > custom > pgwidget > customWidget > dijit > templates`
2. Define a dojoAttachPoint to display the widget attributes on the content pane page.
   a. Open the `CustomWidgetContentPane.html` file.
   b. On line 6, add a dojoAttachPoint with a label "`displayAttributes`" as shown in the following code:

      ```
      <div dojoAttachPoint="displayAttributes" style="width:100%;"></div>
      ```

   c. Save the changes and close the file.
3. Open the `CustomWidgetContentPaneEventListener.js` file in the `ICMCustomWidgets > WebContent > icm > custom > pgwidget > customWidget` folder.

   You use the dojoAttachPoint (`displayAttributes`) that you added to display the string, integer, and boolean widget attribute values.

   a. Observe that in line 25, a variable with a name: `props` is defined and contains code to display each property as shown in the following screen capture.

```
var props = 'String property: ' +
this.contentPane.widgetProperties['customProperty1']
+'<br> Integer property: '+
this.contentPane.widgetProperties['customProperty2']
+'<br> Boolean property: '+
this.contentPane.widgetProperties['customProperty3'];
```

4. In the next line (26), assign the `props` variable to the dojoAttachPoint (`displayAttributes`) as shown in the following line:

   ```
   this.contentPane.displayAttributes.innerHTML = props;
   ```

5. Save the changes and close the file.

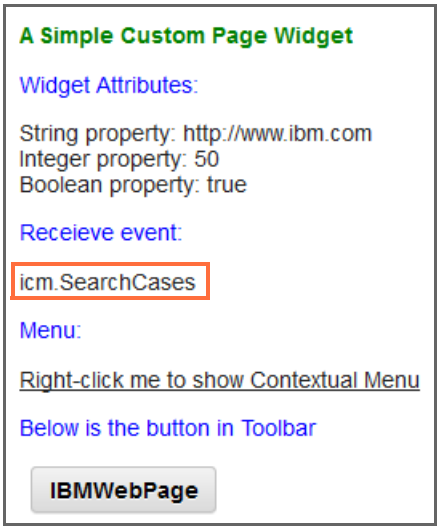
Hint

You can refer to the completed file at `C:\ICM\Widgets\CustomPageWidget\Solution Files\CustomWidgetContentPane-Solution.html`.

Procedure 3: Handle and expose an event

In this procedure, you handle an event that is wired to the custom widget, and expose the event name to the content pane of the custom widget. The event name is shown in your custom widget in Case Manager Client.

For this project, you wire the Search widget to the custom widget and the icm.SearchCases event is displayed as shown in the following screen capture.



1. If it is not already expanded, in Eclipse > Package Explorer > your project, expand `ICMCustomWidgets > WebContent > icm > custom > pgwidget > customWidget > dijit > templates`
2. Define a dojoAttachPoint to display the event name in the content pane page.
   a. Open the `CustomWidgetContentPane.html` file.
   b. On line 8, add a dojoAttachPoint with a label "`displayEvent`" as shown in the following code:

      ```
      <div dojoAttachPoint="displayEvent" style="width: 100%;"></div>
      ```

   c. Save the changes and close the file.
3. Open the `CustomPageWidget.js` file in the `ICMCustomWidgets > WebContent > icm > custom > pgwidget > customWidget` folder.
   a. Starting in line 74, verify that the name of the handler function that was defined earlier in the widget definition JSON file. The name in both files must match.
   b. Call the `displayPayload` function of the CustomWidgetContentPaneEventListener.js file. You defined this Listener with Dojo AMD loading. Add the following code in line 79.

      ```
      this.contentPaneEventListerner.displayPayload(payload);
      ```

   c. Save the changes and close the file.
4. Open the `CustomWidgetContentPaneEventListener.js` file in the `ICMCustomWidgets > WebContent > icm > custom > pgwidget > customWidget` folder.
   a. Locate the `displayPayload` function block on line 19.
   b. Assign the eventName field of the payload to the dojoAttachPoint (`displayEvent`) that you defined earlier. Add the following code in line 20.

      ```
      this.contentPane.displayEvent.innerHTML = payload.eventName;
      ```

   c. Save the changes and close the file.

Hint

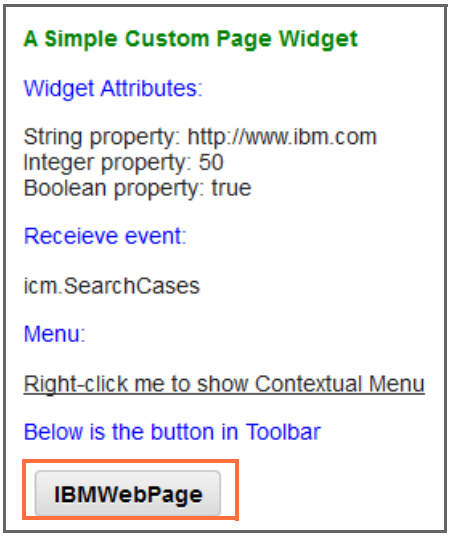
You can refer to the completed file at `C:\ICM\Widgets\CustomPageWidget\Solution Files\CustomWidgetContentPaneEventListener-Solution.js`.

Procedure 4: Incorporate a toolbar button into the custom widget

In this procedure, you incorporate a toolbar action into the custom widget that can be displayed as a button within a toolbar area on the custom widget.

- You defined "open a web page" for the toolbar action in your widget definition file.
- In Case Manager Client, you can access the configuration page from the Page Designer, and add more toolbar action items during the design time.

For this project, the custom widget shows the toolbar button in Case Manager Client as shown in the following screen capture:



1. Define a dojoAttachPoint for the toolbar action for the content pane page.
   a. Open the `CustomWidgetContentPane.html` file.
   b. On line 13, add a dojoAttachPoint with a label "`wrapTopToolbar`".
   c. Set the data-dojo-type to "`dijit.layout.ContentPane`".
   d. Set the `data-dojo-props` to be "`region:'bottom', style: 'border:none;'`".
   e. The completed code looks like the following lines:

```
<div
   data-dojo-type="dijit.layout.ContentPane"
   data-dojo-attach-point="wrapTopToolbar"
   data-dojo-props="region:'bottom',style:'border:none;'">
</div>
```

   f. Save the changes and close the file.
2. Open the `CustomPageWidget.js` file.
   a. Observe the code in line 36. Verify that the toolbar with the dojoAttachPoint (that you defined in the previous step) is instantiated.

```
dojoAttachPoint: "customToolbar"
```

Note

Recollect that "`customToolbar`" is the value for the "`id`" field in the widget definition JSON file that you added in a previous exercise.

   b. Set the toolbar as a content of the page widget to get the action configuration from the page widget in line 39.

```
this.wrapTopToolbar.set("content",this.topToolbar.domNode);
```

   c. Verify that the toolbar is activated in line 42 (`this.topToolbar.startup();`).

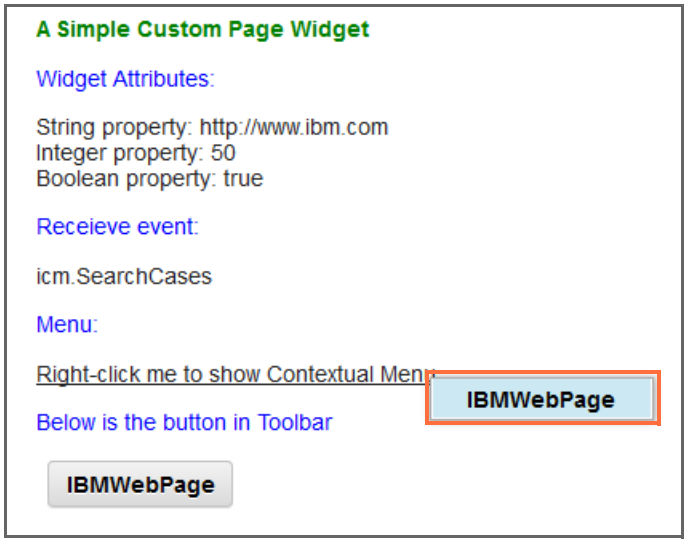
3. Save the changes and close the file.

Procedure 5: Incorporate a menu action into the custom widget

In this procedure, you incorporate a menu action into the custom widget that can be started when the user right-clicks on an area on the custom widget.

- You defined “open a web page” for the menu action in your widget definition file.
- In Case Manager Client, you can access the configuration page from the Page Designer, and add more menu action items during the design time.

For this project, the custom widget shows the menu item in Case Manager Client as shown in the following screen capture:



1. Define a dojoAttachPoint to display for the menu action for the content pane page.
   a. Open the `CustomWidgetContentPane.html` file.
   b. On line 10, check a dojoAttachPoint with a label “`contextualMenuStore`” as shown in the following code:

      ```
      <div data-dojo-attach-point="contextualMenuStore"></div>
      ```

   c. Close the file.
2. Open the `CustomPageWidget.js` file in the `customWidget` folder.
   a. Observe the code in line 48. Verify that the contextualMenu with the dojoAttachPoint (that you defined in the previous step) is instantiated.

      `dojoAttachPoint: “customContextualMenu”`

Note

Recollect that "customContextualMenu" is the value for the "id" field in the widget definition JSON file that you added in a previous exercise.

b. Append the menu in the custom widget to start the menu action configuration from the page widget in line 52.

```
this.contextualMenuStore.appendChild(this.menu.domNode);
```

c. Notice that the target reference of the contextualMenu is set in line 55 so that it can bind to the target point.

d. Verify that the menu is activated in line 61 (`this.menu.startup();`).

3. Save the changes and close the file.
4. Optionally, open the files in the `ICMCustomWidgets > WebContent > icm > custom > action` folder and check the contents. These files implement custom actions.

LESSON 3.6: Build and deploy a widget as an EAR file

What this lesson is about

This lesson describes how to create a widget package that contains an EAR file and register the package.

What you should be able to do

After completing this lesson, you should be able to:

- Create a widget package that contains an EAR file
- Deploy and register a widget package.

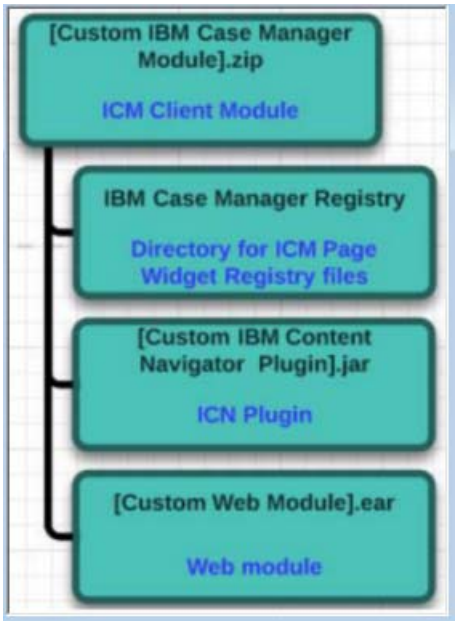
How you will check your progress?

- Hands on labs.

Widget package structure

Contents in a custom widget package

The following diagram shows the files that are included in the ZIP file for the widget package.



Build XML files that are used for this lesson to create the package

This project requires three build xml files to complete the following tasks:

- Create a Jar file for the IBM Content Navigator plug-in (ICMCustomPlugin))
- Create an EAR file for the widget (ICMCustomWidgets) package.
- Creates a final ZIP file that is needed for deployment. The ZIP file contains the following items:
  - JAR
  - EAR
  - Registry folder (ICMRegistry) that contains widget definition and catalog files.

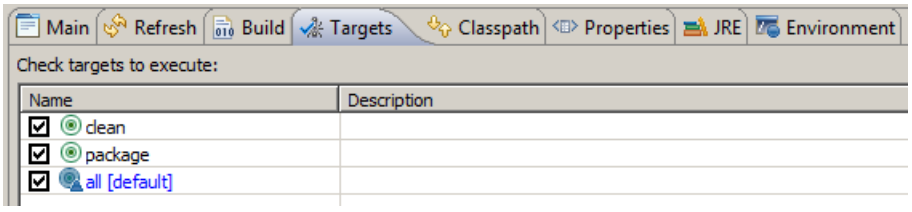
Exercise 3.6.1: Build and deploy the widget package

Procedures

Procedure 1: Create your custom widget package

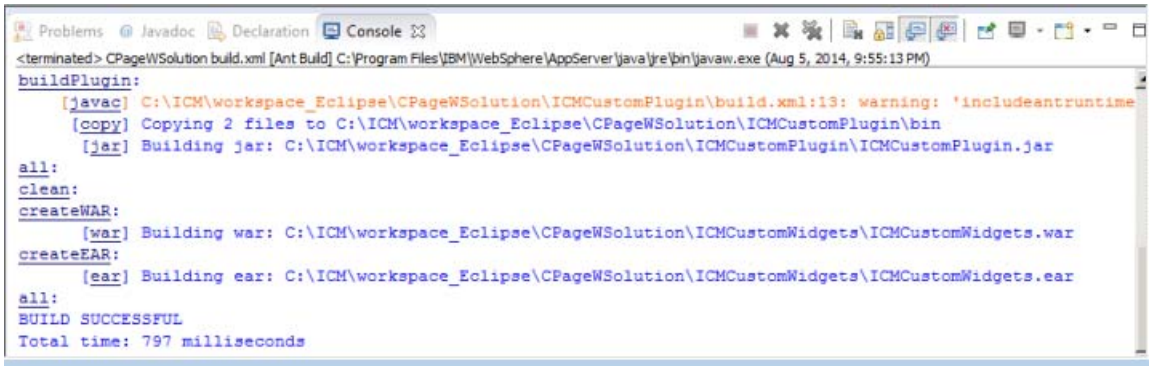
The build xml files that packages your project into a ZIP file is included in the student system. In this procedure, you review the files and generate the package.

1. In Package Explorer, expand your project.
   a. Open the `build.xml` file in the Build folder of your project.
   b. Notice that this file calls the other two build files to create the JAR and EAR files in the `<target name="package">` block.
   c. It creates a final ZIP file that contains the JAR and EAR files and the ICMRegistry folder.
2. Optionally, open the `buildJAR.xml` file in the ICMCustomPlugin folder of your project.
   a. The contents are similar to the file in Lesson 4.
3. Open the `buildEAR.xml` file in the ICMCustomWidgets folder of your project.
   a. Notice that this file creates a WAR file first in the `<target name="Crea...AR">` block.
   b. Then, it creates the EAR file in the `<target name="CreateEAR">` block.
   c. Close all the files.
4. Create a package:
   a. Right-click the `build.xml` in the `Build` folder.
   b. select Run As > "2.Ant Build" from the list.
   c. Make sure to select "all [default]", "clean", and "package", then select Run.

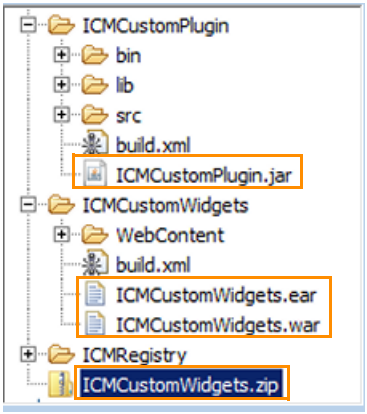


5. Validate that the console tab at the bottom pane displays that the Build was successful.
6. Verify that the message indicates that the JAR, WAR, and ZIP files are created.

7. Ignore the “includeantruntime” warning.



8. Right-click your project and select Refresh from the list.
   a. Verify that the JAR, EAR, and ZIP files are listed in your project.



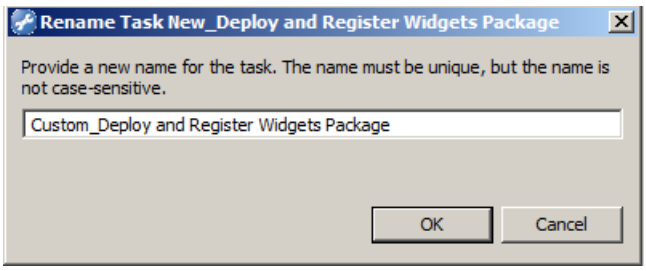
Procedure 2: Deploy and register the widget

In this procedure, you deploy and register the widget in IBM Case Manager configuration tool.

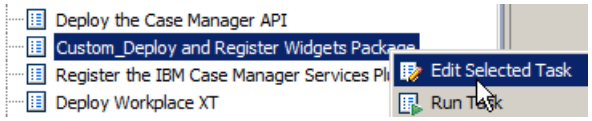
1. Double-click the Case Manager Configuration Tool icon on the desktop.
   a. You can also start it from Windows Start > All Programs > IBM Case Manager > Case Manager Configuration Tool.
2. Select File > Open Profile to open the configuration profile.
   a. In the open window, go to the `C:\Program Files (x86)\IBM\CaseManagement \configure\profiles\ICM Lab` folder.
   b. Select `ICM Lab.cfgp` and click Open.
3. Make a copy of the “Deploy and Register Widgets Package” task.
   a. Expand the ICM Lab node.
   b. Right-click on any task, and select Add New Task > Deploy and Register Widgets Package.

This step makes a copy of the Deploy and Register Widgets Package.

4. Rename the newly created task.
   a. Right-click the “New_Deploy and Register Widgets Package” and select “Rename Task”.
   b. Edit the name to “Custom_Deploy and Register Custom Widgets” and click OK.



5. Edit the task.
   a. Right-click the “Custom_Deploy and Register Widgets Package” and select “Edit Selected Task” from the list.

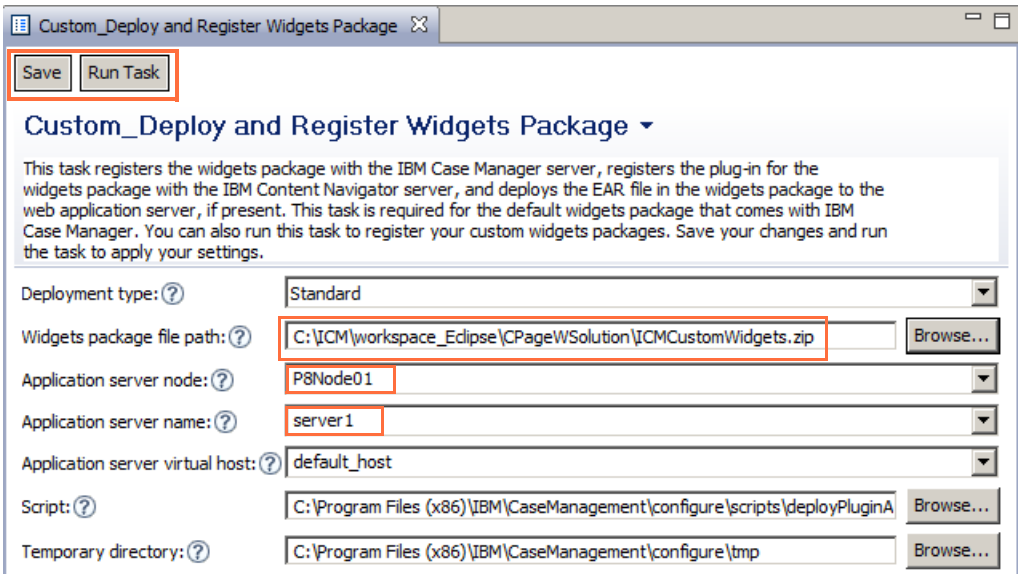


   b. Select the values for the fields with the data in the table. Leave the default values for the other fields.

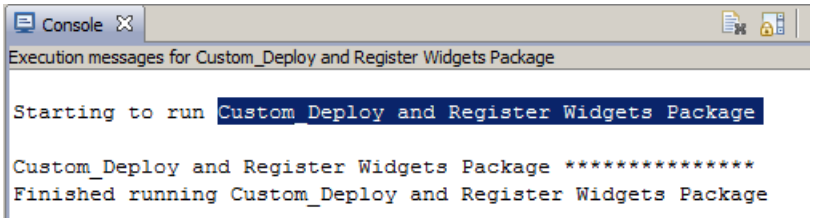
Item	Value
Widgets package file path	C:\ICM\workspace_Eclipse\CPageW\ICMCustomWidget.ZIP
Application server node	P8Node01
Application server name	server1

   c. For “Widgets package file path”, click Browse and select your widget package.

d. Click Save to save your changes.



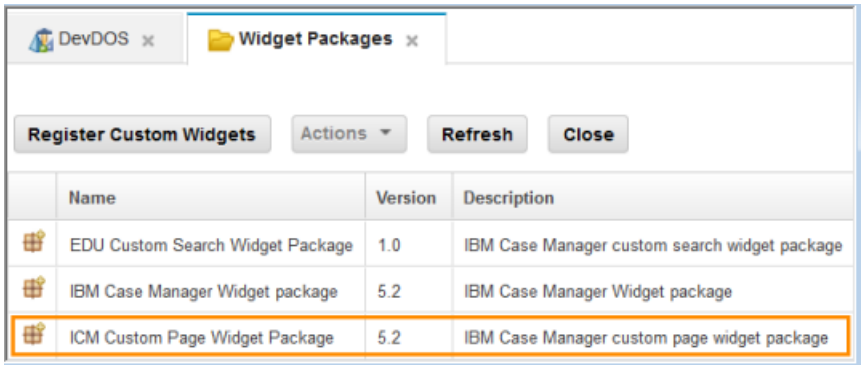
6. Run the task. It takes a few moments.
   a. Validate that the task ran successfully in the console below the task.



7. Click File > Exit to close the IBM Case Manager configuration tool.

Procedure 3: Verify the custom widget package deployment and registration

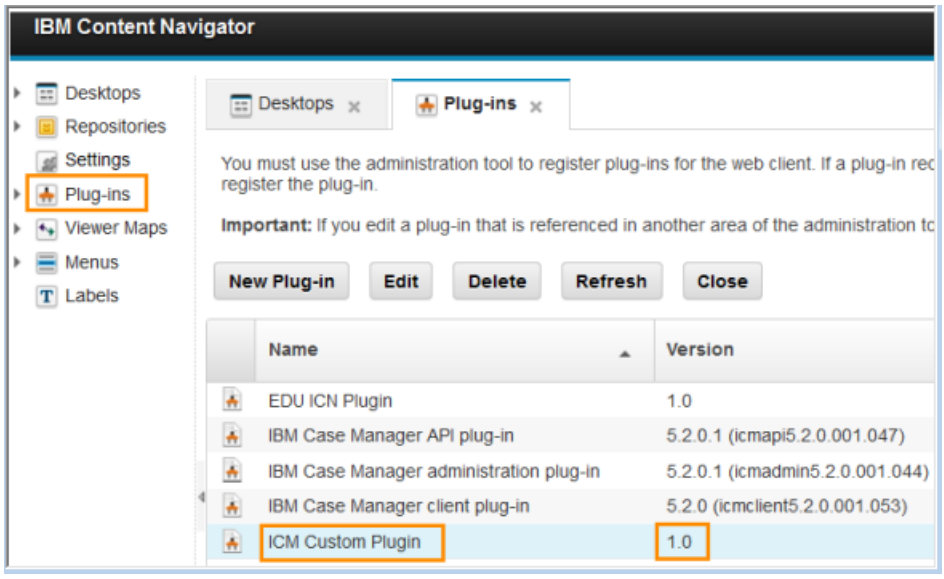
1. Verify that the custom widget package is registered:
   a. Start the IBM Case Manager administration client.
      – URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
      – User name: `P8Admin`
      – Password: `IBMFileNetP8`
   b. Click P8Domain > Object Stores > DevDOS in the left pane to open the Design Object Store.
   c. In the DevDOS tab > left pane, expand the DevDOS and click Widget Packages.
   d. Verify that your widget package is listed in the “Widget Packages” tab.



Note

The catalog and widget definition JSON files are deployed in the `C:\Program Files (x86)\IBM\CaseManagement\configure\properties\widgetsPackage\DevDOS\ICM Custom Page Widget Package` directory.

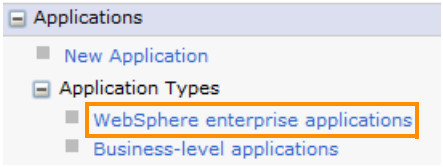
2. Verify that the custom plug-in is registered:
   a. Start the IBM Content Navigator Admin desktop.
      – URL: `http://ecmedu01:9080/navigator/?desktop=admin`
      – User name: `P8Admin`
      – Password: `IBMFileNetP8`
   b. In the IBM Content Navigator Admin desktop, select Plug-ins in the left pane.
   c. In the Plug-ins tab, verify that your plug-in is listed.



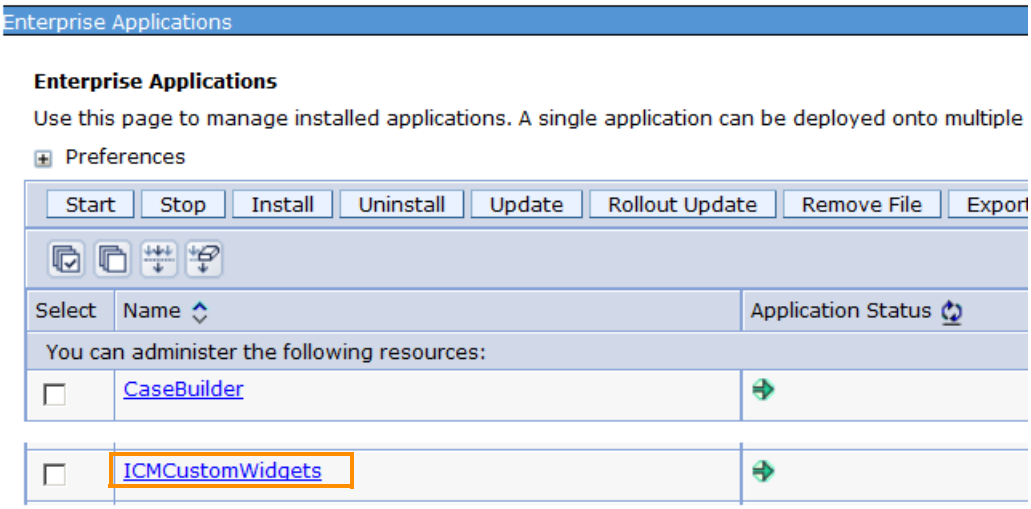
Note

The Content Navigator plug-in for your widget is deployed in the `C:\Program Files (x86)\IBM\CaseManagement\configure\properties\plugins\ICMCustomPlugin.jar` directory.

3. Verify that the widget EAR file is deployed:
   a. Start the WebSphere Integrated Solutions Console.
      – URL: `http://ecmedu01:9043/ibm/console/logon.jsp`
      – User name: `P8Admin`
      – Password: `IBMFileNetP8`
   b. Expand Applications > Application Types and select WebSphere enterprise applications.



   c. Verify that the custom widget EAR file is deployed and `ICMCustomWidgets` is listed.



Note

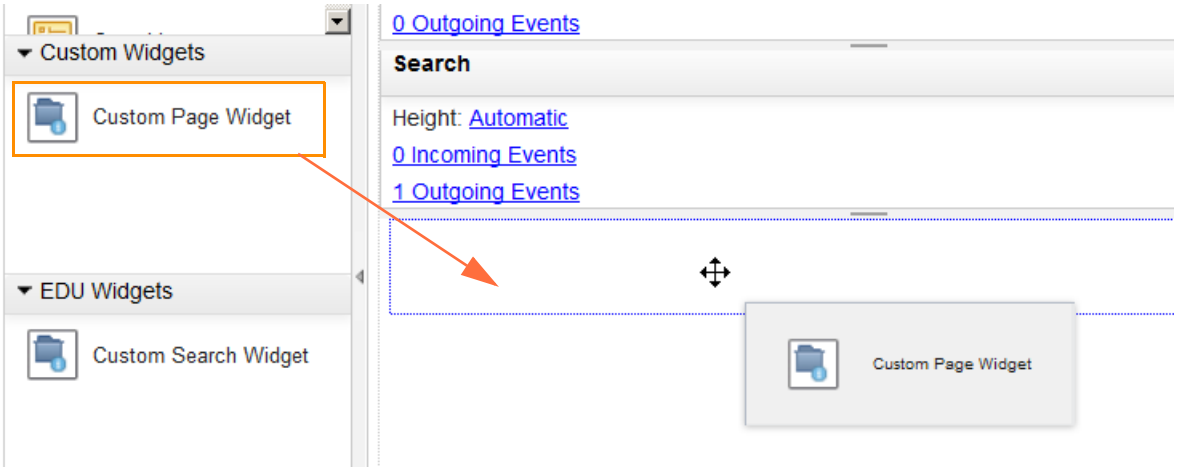
The EAR file for your widget is deployed in the `C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\InstalledApps\P8Node01Cell\ICMCustomWidgets.ear` directory.

   d. Log out of all applications and close the browser.

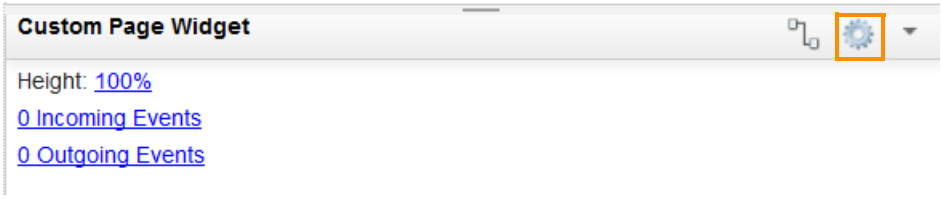
Procedure 4: Create a custom page to add the widget

The steps to test the custom widget are similar to the steps in Lesson 4.

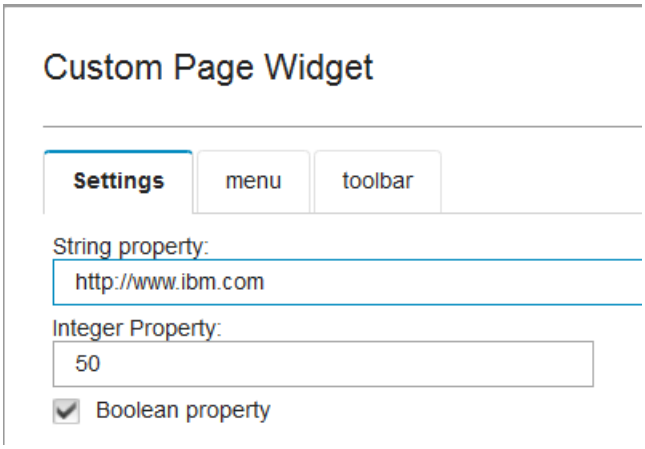
1. In Firefox, log on to IBM Case Manager Builder as an administrative user.
   - URL: `http://ecmedu01:9080/CaseBuilder`
   - User name: `P8Admin`
   - Password: `IBMFileNetP8`
2. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.
   Hover the mouse over the solution to see the links.
3. Create a custom page:
   a. Open the Pages tab and expand the Solution Pages.
   b. Hover the mouse over the Cases page name.
   c. Select the Copy icon on the right side of the page.
   d. In the resulting page, edit the name to `Custom Page` for your new page and click OK to create the copy.
   e. Save your work by clicking Save at the top of the page.
4. Edit the page in Page Designer to add the custom widget.
   a. In the Pages tab, double- click `Custom Page`.
   b. Drag your “Custom Page Widget” (under the “Custom Widgets” section) from widget palette on the left column to the page on the right.
   c. Place the widget below the existing default Search widget.



5. Optionally, configure the properties.
   a. Click the “Edit Settings” icon of the Custom Page Widget.



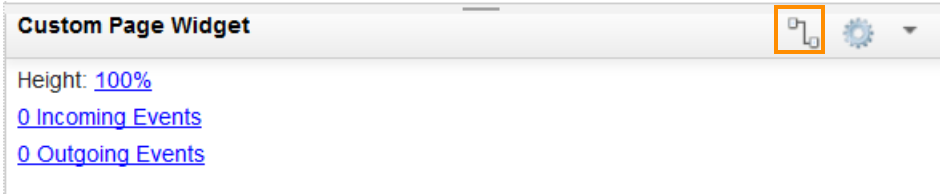
b. In the settings tab, enter values for the properties fields as shown in the screen capture.



c. Click OK to close the page.

6. Edit the wiring for the widget:

a. Click the “Edit Wiring” icon of the Custom Page Widget.

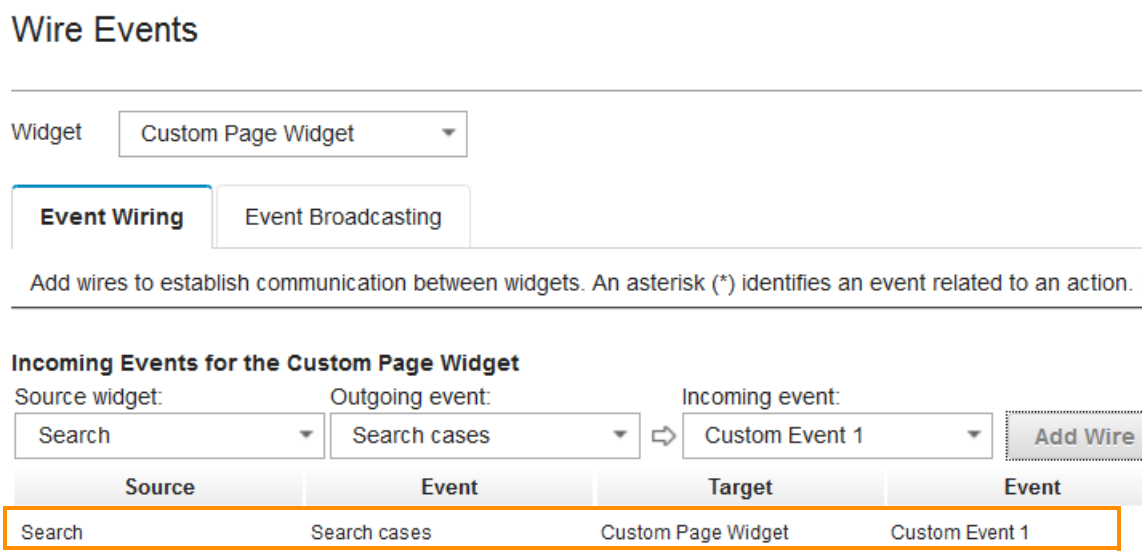


b. In the Wire Events page > Event Wiring tab > “Incoming Events for the Custom Page Widget” section, complete the wiring with the data in the following table.

Field	Value
Source widget	Search
Outgoing event	Search cases
Incoming event	Custom Event 1

c. Click Add Wire.

d. The completed page looks like the one in the following screen capture.



e. Click OK to close the page.

f. Click Save and then Close to save your work and close Page Designer.

7. Assign the custom page to a role.
   a. Open the Roles tab.
   b. Click the Customer Service Rep role link.
   c. Open the Pages subtab.
8. Remove the Custom Search page that you created in the previous lesson.
   a. Select the page, hover over, and click the Remove (trash can) icon.
9. Assign the new page that you created.
   a. Click Assign Page.
   b. Select `Custom Page`.
   c. Click OK to close the dialog window.
10. Verify that your page is listed in the Pages tab.
    a. Click OK and then “OK All” to accept the changes to the role.
    b. Click “Save and Close” at the top of the page to exit the solution editor.
11. Redeploy the solution.
    a. In the Manage Solutions page, commit the changes to the Lab Claims Solution and Deploy it.
    b. Wait for the green check mark to appear next to the solution.
12. Hover the mouse over the solution and click Test to open Case Manager Client.

Procedure 5: Test the custom widget

1. In the Case Manager Client, select the Custom page tab to open your custom page.
2. Verify that the custom widget is displayed below the Search on the left pane.
   a. Under the “Widget Attributes” section, a list of properties that you configured in the widget definition file is shown.
3. Test the menu.
   a. Right-click the menu and check that the menu is shown.
4. Verify that the toolbar button is shown.
   a. Click the “IBMWebPage” button, and check that it opens the web page.
5. Verify that the event is displayed.
   a. Click the Search button to create a “Search Cases” event.
   b. Verify that the custom widget shows the `icm.SearchCases` event.

6. Logout of the applications and close the browser.

Exercise 3.6.2: Troubleshooting

Introduction

Troubleshooting

If you get errors when you test your custom widget in the Case Manager Client, do the following steps.

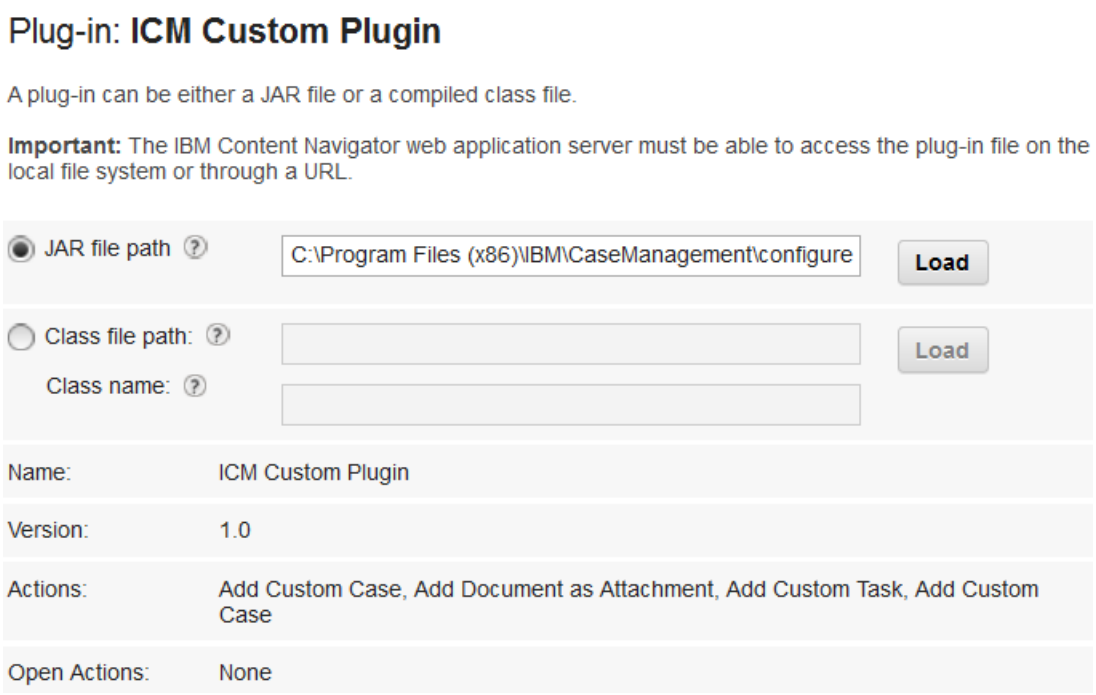
1. Check your code and if needed, replace your code with the solution files.
2. Package the code again into a ZIP file.
   a. Refer to Procedure 1: Create your custom widget package, page 78 for more details.
3. Redeploy and register your new package.
   a. Refer to Procedure 2: Deploy and register the widget, page 79 for more details.
4. Optionally, verify the deployment and registration.
   a. Refer to Procedure 2: Register the custom widget, page 46 for more details.
5. Verify the Content Navigator plug-in.
   a. Refer to Procedure 2: Verify the Content Navigator plug-in, page 55 for more details.
6. Clear the browser cache.
   a. The student image is configured to clear the cache when you close the browser.
7. Test your widget.
   a. Refer to Procedure 5: Test the custom widget, page 87 for more details.

Procedure 1: Verify the Content Navigator plug-in

In this procedure, you verify that the plug-in that you developed is registered properly in the IBM Content Navigator admin tool.

1. Start the IBM Content Navigator Administration Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=admin`
   - User name: `P8Admin`
   - Password: `IBMFileNetP8`
2. Open the Plug-ins tab.
   a. Select Plug-ins in the left pane.
3. In the Plug-ins tab, select your plug-in (`EDU ICN Plugin`) and click Edit.

4. In the `ICM Custom Plugin` tab, click Load for the JAR file path.
   a. Verify that the details about the plug-in is displayed as shown in the following screen capture.



5. Click Save and Close.
6. Log out of the IBM Content Navigator Administration Desktop and close the browser.

LESSON 3.7: Update an existing package with new widgets

What this lesson is about

This lesson describes how to create a case comments widget, and update an existing widget package with new widgets. You can use the similar steps to update an existing widget in the package.

What you should be able to do

After completing this lesson, you should be able to:

- Create a case comments widget.
- Update an existing package with a new widget.

How you will check your progress?

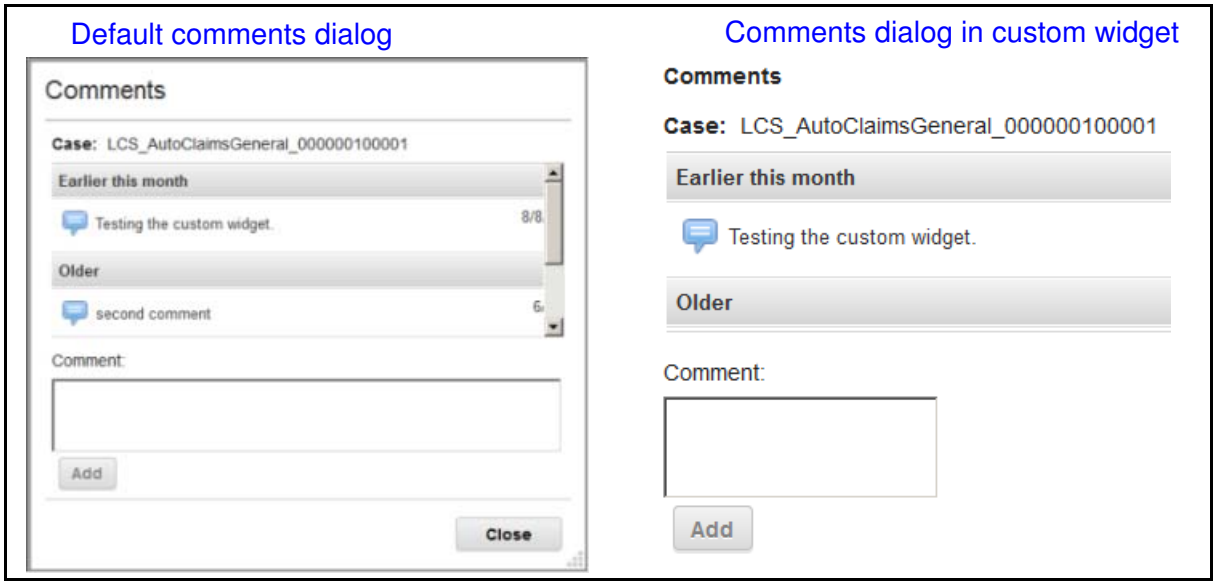
- Hands on labs.

Custom case comment widget

Default comments dialog dijit

The following screen capture shows the comments dialog interface both in the default IBM Case Manager Client and in the custom widget that you develop in this lesson.

- The default dialog opens, when you click the Comments button in the Case Details page.
- You add your custom widget in the Case Details page, and the comments dialog is available when you open the case in Case Details page.



- The custom case comment page widget (`icm/custom/pgwidget/commentWidget/CommentWidget.js`) that is used in this lesson embeds the IBM Case Manager-provided comment dialog dijit.

- At runtime, the case comment page widget coordinates with case toolbar page widget to save the unsaved case comment automatically when you save the case.

Update a custom widget package

Update an existing custom widget

If you update the code for your custom widget, use the following steps to redeploy the custom widget. Some of the steps are optional while others are mandatory.

1. In Case Manager Builder > Page Designer, remove the custom widget from all the pages where you added the widget. (optional)
2. After you update and package the widget, run the “Deploy and Register Widgets” task in IBM Case Manager Configuration Tool with the updated ZIP file for the custom widget. (mandatory)*
3. Clear the browser cache and cookies. (mandatory)*
4. In Case Manager Builder > Page Designer, add the new custom widget to the pages (optional: do this step only if you did the step 1)
   a. Commit the changes and deploy the solution with your new widget.(optional)

Important

Steps 2-3 are sufficient for simple changes to the widget JavaScript files such as message outputs. If you change the widget definition JSON file that includes events, or wiring of the widget, then you must do all steps.

Update an existing custom widget package with new widgets

If you add a widget to an existing widget package, use the following steps to redeploy the custom widget package.

1. In Eclipse (or any other development tool), create and add a widget to your existing project.
   a. Build the project to create a final ZIP file.
2. Run the “Deploy and Register Widgets” task in IBM Case Manager configuration tool with the updated ZIP file for the new custom widget. The tasks update the following items:
   – The web application in WebSphere Application Server
   – Registration for the custom widget package
   – The IBM Content Navigator plug-in
3. In Case Manager Builder > Page Designer, add the new custom widget to a page.
   a. Commit the changes and deploy the solution with your new widget.

Exercise 3.7.1: Update an existing package with new widgets

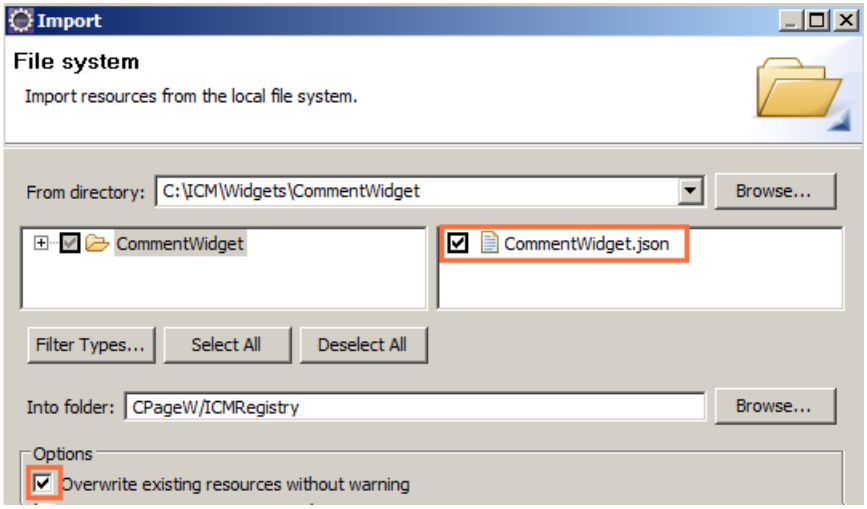
Introduction

In this exercise, you create a comment widget, and add it to your existing package that you created in the previous lesson. You must redeploy the updated widget package to be able to use the new widget.

Procedures

Procedure 1: Create registry files for the custom widget

1. If it is not already opened, start Eclipse by double-clicking the Eclipse icon in your desktop.
   a. In the Workspace Launcher page, leave the default workspace directory (`C:\ICM\workspace_Eclipse`) and click OK.
2. Import the CommentWidget.json file into your project.
   a. In Eclipse > Package Explorer > CPageW, right-click the `ICMRegistry` folder and click Import from the list.
   b. In the Import page, expand General, select “File System”, and click Next.
   c. In the Import > File system page, click Browse.
   d. In the “Import from directory” page, expand `C:\ICM\Widgets` folder, select the `CommentPageWidget` folder, and click OK.
   e. Back in the Import > File system page, select `CommentWidget.json` in the right pane.
   f. Make sure that the “`Into folder`” field has the following value: `CPageW/ICMRegistry`
   g. Select the “Overwrite existing resources without warning” option.
   h. Click Finish.



3. Open the `CommentWidget.json`.
   a. The contents of the file is similar to the widget definition file of page widget that you created in the previous lesson.
   b. Check that it has “properties” and “events” section.

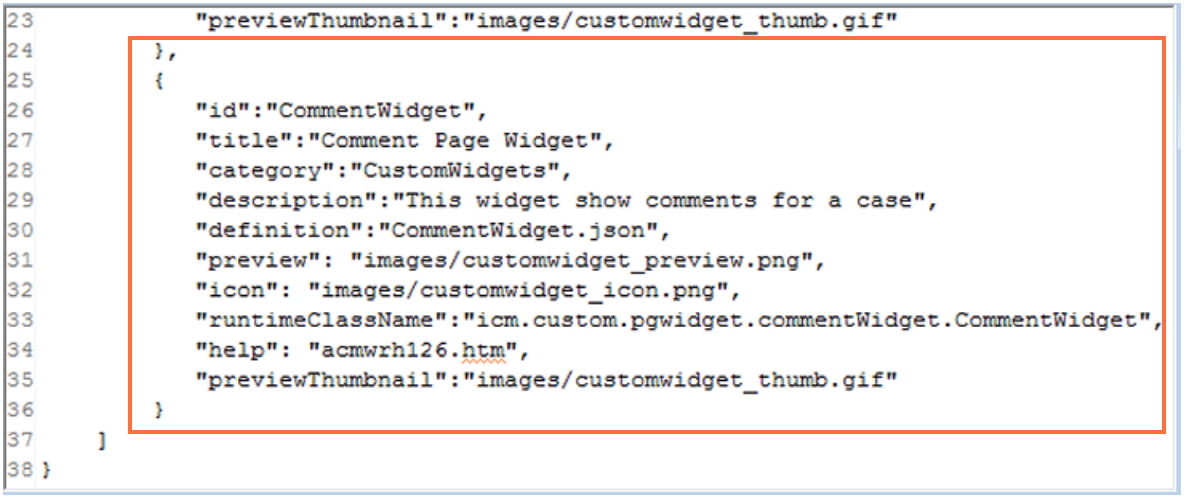
      This widget handles two events: Select case and Send case information.

      The two methods that are specified in this file (handleICM_SelectCaseEvent and handleICM_SendCaseInfoEvent) are defined in the JavaScript file.

```
"events":[
    {
        "id": "icm.SelectCase",
        "title": "Select case",
        "functionName": "handleICM_SelectCaseEvent",
        "direction": "subscribed",
        "description": "Display the case information for the case
    },
    {
        "id": "icm.SendCaseInfo",
        "title": "Send case information",
        "functionName": "handleICM_SendCaseInfoEvent",
        "direction": "subscribed",
        "description": "Display the case that is specified in the
    }
```

   c. Copy the lines 2-11 to paste it in the catalog file.
4. Edit the `Catalog.json` file to include the `CommentWidget`.
   a. Open the `Catalog.json` file that you added for the custom page widget.
   b. Check that in the “Widgets” section, you have a block for “CustomPageWidget”.
   c. In line 24, after the closed curly braces (before the square bracket) add a comma.

d. In the next line, add an open curly brace.
e. Paste the lines that you copied from the `CommentWidget.json`.
f. In line 36, add a closed curly brace (before the square bracket).
g. The completed code must look like the following screen capture. The required code is highlighted.

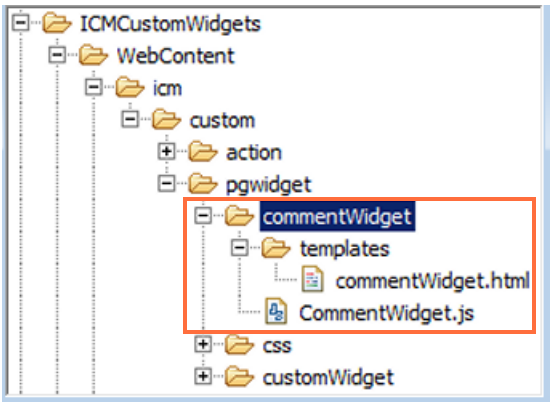


```
23              "previewThumbnail":"images/customwidget_thumb.gif"
24          },
25          {
26              "id":"CommentWidget",
27              "title":"Comment Page Widget",
28              "category":"CustomWidgets",
29              "description":"This widget show comments for a case",
30              "definition":"CommentWidget.json",
31              "preview": "images/customwidget_preview.png",
32              "icon": "images/customwidget_icon.png",
33              "runtimeClassName":"icm.custom.pgwidget.commentWidget.CommentWidget",
34              "help": "acmwrh126.htm",
35              "previewThumbnail":"images/customwidget_thumb.gif"
36          }
37      ]
38  }
```

Procedure 2: Implement the CommentWidget

The files that are required for the CommentWidget are included in the student system. In this procedure, you copy these files into your project.

1. Copy the folders for your project.
   a. In Windows Explorer, go to the `C:\ICM\Widgets\CommentPageWidget` folder.
   b. Right-click the `commentWidget` folder and select Copy.
   c. In Eclipse > Package Explorer > CPageW, expand `ICMCustomWidgets > WebContent > icm > custom > pgwidget`
   d. Right-click `pgwidget` and select Paste.
   e. Expand the `commentWidget` folder to see the contents.
   f. The directory must look like the following screen capture:



Procedure 3: Build and deploy the widget package

In this procedure, you build and deploy the widget package to make your new widget available in the IBM Case Manager.

1. Build the widget package (as a Zip file)
   a. See the Procedure 1: Create your custom widget package, page 78.
2. Deploy and register the widget package.
   a. See the Procedure 2: Deploy and register the widget, page 79.
   b. You can reuse the “Custom_Deploy and Register Custom Widgets” task.
3. Verify the deployment and registration.
   a. See the Procedure 2: Register the custom widget, page 46.

Procedure 4: Create a custom page to add the widget

The steps to test the custom widget are similar to the steps in Procedure 1: Create a custom page, page 50.

1. In Firefox, log on to IBM Case Manager Builder as an administrative user.
   - URL: `http://ecmedu01:9080/CaseBuilder`
   - User name: `P8Admin`
   - Password: `IBMFileNetP8`
2. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.
   Hover the mouse over the solution to see the links.
3. Create a custom page:
   a. Open the Pages tab and expand the `Case Details Pages`.
   b. Hover the mouse over the `Case Details` page name.
   c. Select the Copy icon on the right side of the page.

d. In the resulting page, edit the name to `Comments Page` for your new page and click OK to create the copy.

e. Save your work by clicking Save at the top of the page.

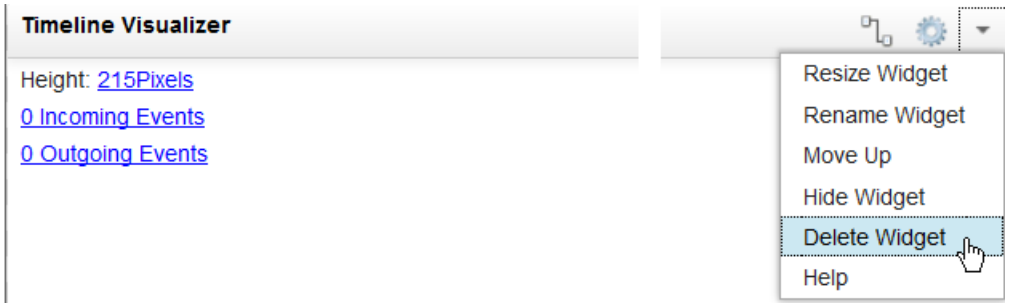
4. Edit the page in Page Designer to add the custom widget.

a. In the Pages tab, double- click `Comments Page`.

b. In Page Designer > Comments Page, remove the Timeline Visualizer widget to make room for your widget.

c. Click the down-arrow in the toolbar of your widget, and select “Delete Widget”.

d. If the toolbar controls are not visible, expand the area for the widget.

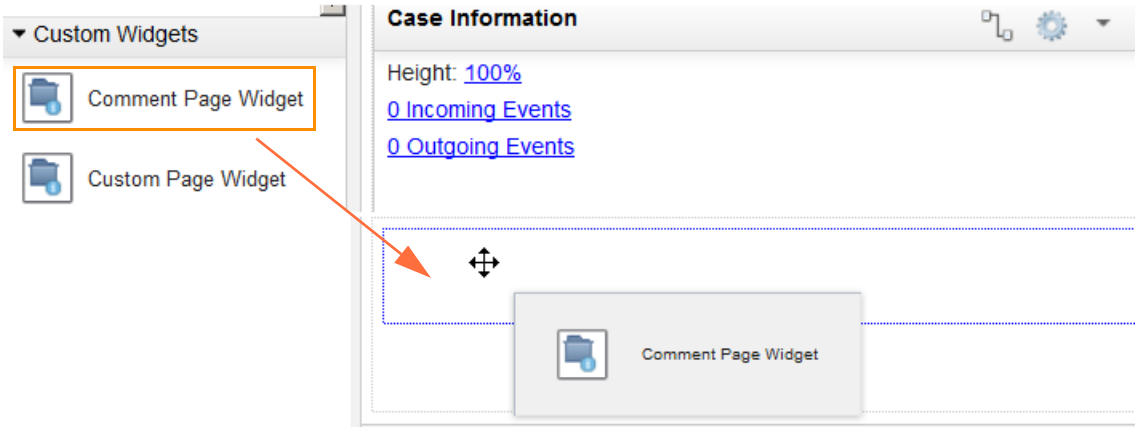


5. Add Comment Page Widget to the page.

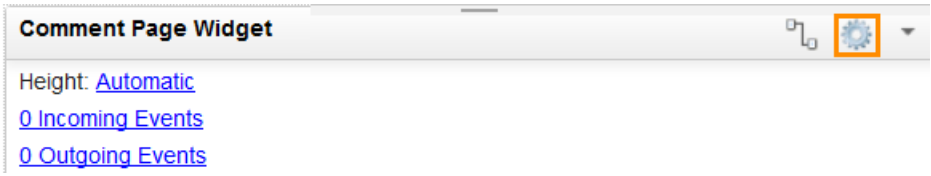
a. Verify that the widget is added to the Custom Widgets category in the widget palette on the left pane.

b. Drag “Comment Page Widget” to the page.

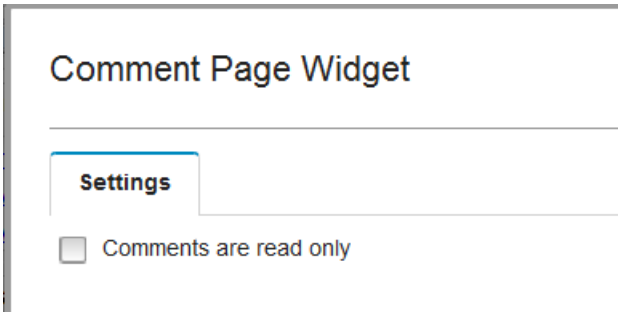
c. Place the widget below the existing Case Information and Properties widgets.



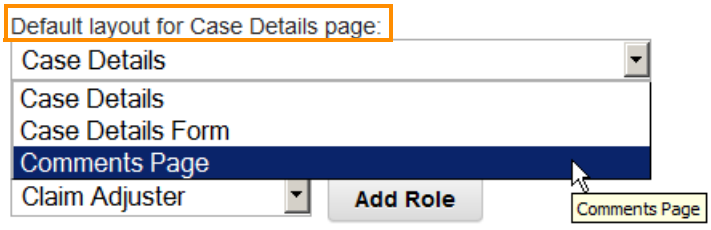
6. Check the configuration page for Comments Page Widget.
   a. Click the “Edit Settings” icon.



   b. In the settings tab, notice the option to make comments “read only” as shown in the screen capture.



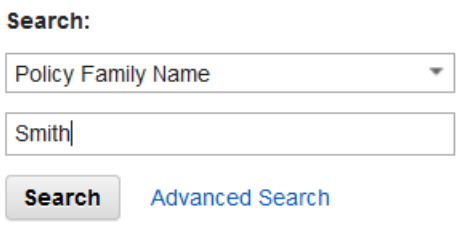
   c. Click OK to close the page.
7. Click Save to save your changes and Close to close Page Designer.
8. Assign the new page to a case type.
   a. Select the Case Types tab.
   b. Select Auto Claims General.
   c. In the Case Type page, select Comments Page for “Default layout for Case Details page”.



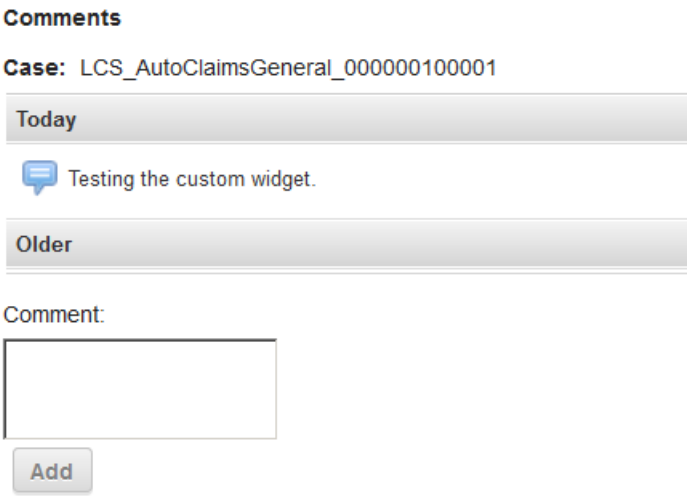
9. Click Save and Close to save the changes to the solution.
10. Redeploy the solution.
    a. In the Manage Solutions page, commit the changes to the Lab Claims Solution and Deploy it.
    b. Wait for the green check mark to appear next to the solution.
11. Hover the mouse over the solution and click Test to open Case Manager Client.

Procedure 5: Test the custom widget

1. In the Case Manager Client, select the Cases tab.
   a. Do a Search (criteria: Policy Family Name = `Smith`)
      The value is case-sensitive; make sure “S” is uppercase.



2. Click the Title link for the case in the right pane to open the Case details in the Comments Page.
3. In the Comments Page tab, verify that the Comments widget is shown at the end of the page.
   a. Enter any text in the “Comment” input field to add a comment for the case.
   b. Click Add. Verify that the comment is listed.



   c. Click the Comments button at the top of the page. A page similar to the custom Comments widget dialog opens.
      The custom widget implements the IBM Case Manager-provided comment dialog dijit that the default Comments widget also implements.
   d. Click Close to close the Case Details (Comments Page) page.
4. Logout of the applications and close the browser.

Appendix: Uninstall a custom widget in IBM Case Manager

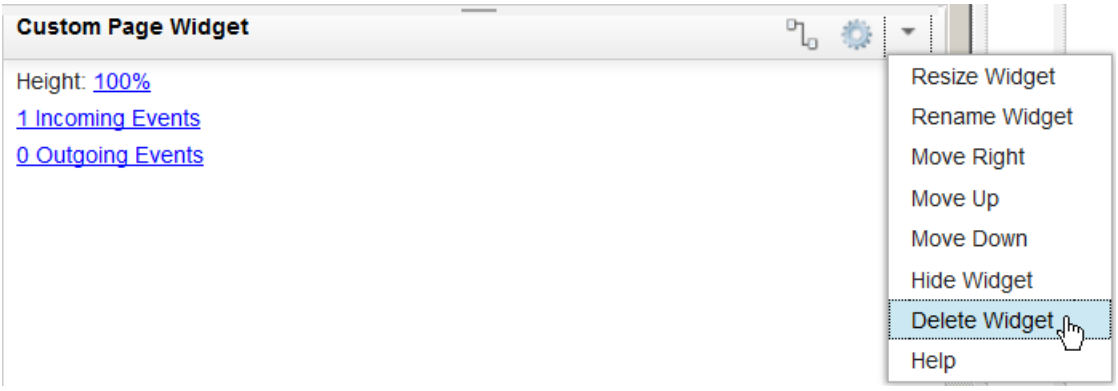
Introduction

If you require to remove the custom widgets from the IBM Case Manager system, you must do the procedures in this section.

Procedures

Procedure 1: Remove the custom widget

1. In Firefox, log on to IBM Case Manager Builder as an administrative user.
   - URL: `http://ecmedu01:9080/CaseBuilder`
   - User name: `P8Admin`
   - Password: `IBMFileNetP8`
2. In Case Manager Builder, open Lab Claims Solution by clicking the Edit link.
   Hover the mouse over the solution to see the links.
3. Open the custom page in Page Designer to edit it:
   a. Select the Pages tab and expand the Solution Pages.
   b. Double-click your custom page (`Custom Page`).
4. In the toolbar of your widget, click the down-arrow and select “Delete Widget”.
   a. If the toolbar controls are not visible, expand the area for the widget.



5. Click Save to save your changes.
   a. Click Close to close Page Designer.
   b. Click Save and Close to save the changes to the solution.

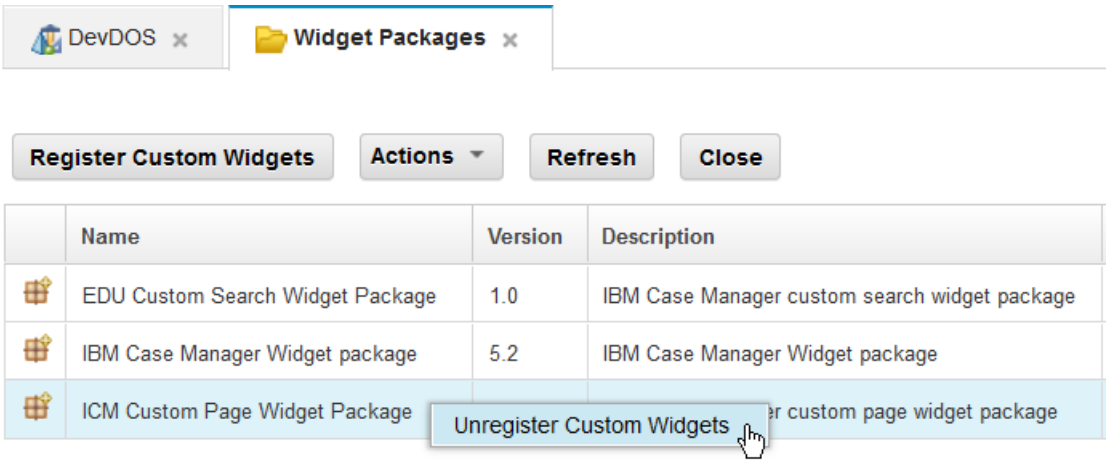
Note

If a page contains multiple instances of the custom widget, you must delete all of them. If multiple pages include the custom widget, you must delete it from each page.

Procedure 2: Unregister the custom widget package

In this procedure, you unregister the widget package in the IBM Case Manager admin tool.

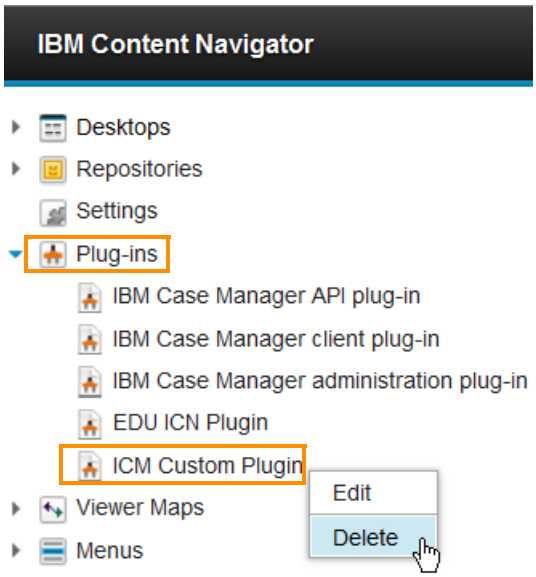
1. Start the IBM Case Manager administration client.
   - URL: `http://ecmedu01:9080/navigator/?desktop=icmadmin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Open the Design Object Store.
   a. Click P8Domain > Object Stores > DevDOS in the left pane.
3. In the DevDOS tab, expand the DevDOS and select “Widget Packages” in the left pane.
   a. In the Widget Packages tab, right-click your widget package and select “Unregister Custom Widgets”.



4. In the “Unregister Custom Widgets” tab, click Finish.
   a. When you get the message that the package is removed, click Close.
   b. Verify that your widget package is removed from the list in the Widgets Packages tab.
5. Log out of the IBM Case Manager admin tool and close the browser.

Procedure 3: Delete the IBM Content Navigator plug-in

1. Start the IBM Content Navigator Administration Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=admin`
   - User name: `P8Admin`
   - Password: `IBMFileNetP8`
2. Expand “Plug-ins” in the left pane.
3. Right-click your plug-in (`ICM Custom Plugin`) and click Delete.
4. Click Delete when you are prompted to confirm.



5. Verify that your plug-in is deleted from the list.
6. Log out of the IBM Content Navigator Administration Desktop and close the browser.

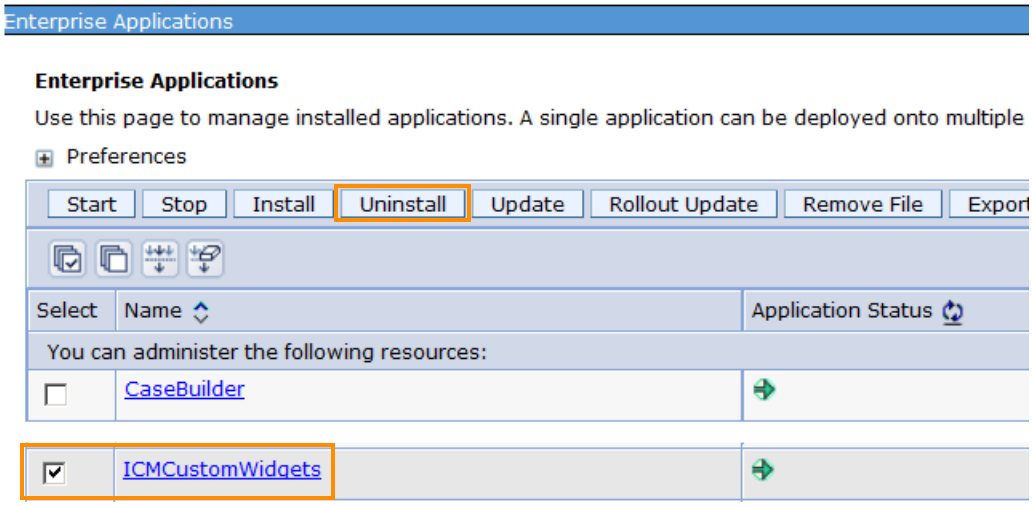
Procedure 4: Uninstall the web application

If the widget package contains a web module for your widgets (EAR file), you uninstall the web application for your custom widget package.

1. Start the WebSphere Integrated Solutions Console.
   - URL: `http://ecmedu01:9043/ibm/console/logon.jsp`
   - User name: `P8Admin`
   - Password: `IBMFileNetP8`

   c. In the left pane, expand Applications > Application Types and select WebSphere enterprise applications.

   d. Select `ICMCustomWidgets` from the list and click Uninstall.



e. In the “Uninstall Application” page, click OK.

f. In the “Enterprise Applications” page, click Save in the Messages section to save the changes to master configuration.

g. Verify that the application for your widget package is removed from the list.

2. Log out of WebSphere Integrated Solutions Console and close the browser.

Important

You must restart the WebSphere Application Server, and clear the browser cache.