IBM Content Navigator 2.0.2: Plug-ins and External Data Services

Course Code: F122
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:

FileNet

IBM

DB2

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.

March 2014 edition

Trademarks
The information contained in this document has not been submitted to any formal IBM test and is distributed on an “as is” basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

TABLE OF CONTENTS

Course Overview

- Course Description 0-2
- Development options with IBM Content Navigator 0-4

Unit 1: Implement External Data Services

Lesson 1.1: Implement External Data Services

- Request and Response filters 1-3
- Request and Response filters implemented in the sample EDS 1-5
- External Data Services (EDS) 1-7
- EDS REST protocol 1-9
- Demonstrations 1-10
- Exercise 1.1.1:Register IBM Content Navigator EDS plug-in 1-11
- Exercise 1.1.2:Implement property field validation 1-14
- Exercise 1.1.3:Set the input field status as required 1-18
- Exercise 1.1.4:Prefill an initial value for a data field 1-26

Lesson 1.2: Create Choice Lists with Sample EDS

- Create choice lists with External Data Service (EDS) 1-30
- Exercise 1.2.1:Add an external choice list with EDS. 1-32
- Exercise 1.2.2:Create dependent choice lists with EDS 1-35
- Exercise 1.2.3:Change the field status dynamically for a property 1-37
- Additional Information. 1-40
- Appendix: Steps to import the sample EDS into Eclipse 1-41
- Appendix: Steps to import the sample EDS into Eclipse 1-44
- Appendix: Steps to import the sample EDS into Eclipse 1-45

Unit 2: Develop Plug-ins

Lesson 2.1: IBM Content Navigator Development Overview

- IBM Content Navigator development architecture 2-3
- Components of IBM Content Navigator 2-5
- Develop applications with IBM Content Navigator 2-7
- Setting up the development environment for plug-ins 2-9
- Demonstrations 2-12
- Exercise 2.1.1:Create a plug-in project in Eclipse. 2-13
- Exercise 2.1.2:Register the plug-in with IBM Content Navigator. 2-19
- Lesson Summary 2-21

Lesson 2.2: Implement Custom Response Filters

- IBM Content Navigator Plug-ins 2-23
- Request and Response filters 2-24
- IBM Content Navigator plug-in extensions 2-25
- Content List and Document Information areas for this lesson 2-26

- Files that are used for this lesson 2-27
- Demonstrations 2-29
- Exercise 2.2.1:Implement custom Response Filters. 2-30
- Lesson Summary. 2-39

Lesson 2.3: Create a Custom Property Editor

- IBM Content Navigator terms that are used in this lesson. 2-41
- Custom Property Editors 2-42
- Files that are used for this lesson 2-43
- Exercise 2.3.1:Add a custom property editor 2-44

Lesson 2.4: Develop a Plug-in to Add a Custom Feature

- Add a feature to IBM Content Navigator interface. 2-51
- Creating client-side plug-in extensions for a feature 2-52
- Files that are used for this lesson 2-53
- Exercise 2.4.1:Develop a plug-in to add a feature 2-54
- Lesson Summary. 2-62

Lesson 2.5: Develop Plug-in to Add an Action

- Develop a plug-in to add a menu action 2-64
- Creating client-side plug-in extensions for an action. 2-65
- Files that are used for this lesson 2-66
- Exercise 2.5.1:Develop a plug-in to add an action 2-67

Unit 3: Create a Custom Workflow Step Processor

Lesson 3.1: Create a Custom Step Processor

- Workflows in IBM Content Navigator. 3-3
- IBM Content Navigator step processors 3-5
- Custom step processors 3-7
- Demonstrations 3-9
- Exercise 3.1.1:Develop a custom step processor. 3-10
- Exercise 3.1.2:Register the custom step processor 3-15
- Exercise 3.1.3:Configure Application Space Roles and In-baskets 3-19
- Exercise 3.1.4:Assign the custom step processor to a workflow 3-22
- Exercise 3.1.5:Modify the Workflow Subscription. 3-25
- Exercise 3.1.6:Add External Data Services to step processor 3-29

Lesson 3.2: Add an Action to the Custom Step Processor

- Add an action to use embedded viewer. 3-34
- Exercise 3.2.1:Add an action to use the embedded viewer 3-36

Unit 4: Use Content Navigator Widgets in Other Applications

Lesson 4.1: Use Content Navigator Widgets in Other Applications

- Externalize IBM Content Navigator widgets 4-3
- Unbound integration 4-4
- Externalize with URL API and deep linking 4-5
- Externalize with a custom Feature. 4-6
- Externalize with a custom Layout 4-7
- Bound integration 4-8
- Demonstrations 4-9
- Exercise 4.1.1:Set up an external web application. 4-10
- Exercise 4.1.2:Integrate Content Navigator with URL API 4-15
- Exercise 4.1.3:Integrate Content Navigator with a Feature 4-19

Unit 5: Appendix

Appendix A: Start and Stop System Components

Appendix B: Debugging and troubleshooting

Appendix C: Develop ECM widgets solutions in IBM Content Navigator

Appendix D: Additional information and links

Course Overview

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

Course Description

Overview

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

Course Objectives

After completing this course, you should be able to:

- Implement External Data Services (EDS).
  - Prefill an initial value for a data field.
  - Implement property field validation.
  - Create dependent choice lists with sample EDS.
  - Change the field status dynamically for a property.
- Develop custom plug-ins to do the following tasks:
  - Implement custom Response Filters.
  - Add a custom Property Editor.
  - Add a custom Feature.
  - Add a custom Action.
- Create a custom workflow step processor.
  - Develop and configure a custom step processor.
  - Add external data services to step processor.
  - Add an action to the custom step processor.
- Use Content Navigator widgets in other applications.
  - Integrate Content Navigator with URL API.

Audience

The intended audiences for this course are:

- Developers who are responsible for
  - Customizing and extending the IBM Content Navigator features by developing plug-ins.
- Anyone who needs to know the capabilities of IBM Content Navigator.

Prerequisites

- Intermediate level of expertise in the following technologies:
  - Java

  - JavaScript
  - JSON
  - HyperText Markup Language (HTML 5)
  - Cascading Style Sheets (CSS3)
  - Dojo (version 1.8.4)
- Experience with Eclipse IDE for developing applications.
- Familiarity with deploying applications in WebSphere Application Server.
- Familiarity with Content and Business Process Management (workflow) concepts
- Recommended Courses:
  - F205 - Introduction to IBM Content Navigator 2.0
  - F121 - IBM Content Navigator 2.0.2: Administration

Development options with IBM Content Navigator

Options

IBM Content Navigator provides the following options to develop the user experience for your application. Each option requires different development skill set and time to achieve the goal.

- Configure IBM Content Navigator
- Implement the External Data Service (EDS) interface
- Develop a plug-in
- Integrate a custom application

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

Configure the IBM Content Navigator desktop appearance

You can change the appearance of the desktops 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 of an IBM Content Navigator desktop.
  - Specify the name of the desktop
  - Add a company logo to both the login page and the banner of the desktop.
  - Alter the colors that are used in the desktop banner.
- Configure visual aspects for all desktops on the global level:
  - Change icons for specific mime-types.
  - Modify the labels that are used in the desktop.
- Configure security and business requirements for the complete solution.
  - Include useful notes to the login page such as “forgot password” information.
  - Add or remove features from the desktop.
  - Alter the menu items on both toolbar and context menus.
  - Specify the properties to show in the user interface:

Note

This topic is covered in the “F121 - IBM Content Navigator 2.0.2: Administration” course.

Implement the External Data Service (EDS) interface

The 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 without the need to understand Java or implement an IBM Content Navigator plug-in.

EDS REST interface implementation enables the following tasks:

- Pre-fill properties with values
- Look up the choice list values for a property or dependent property
- Set minimum and maximum property values
- Set property status or controls, such as read-only, required, or hidden
- Implement property validation and error checking

Required skills

You need the following skills to implement the external data service REST interface:

- – Web application development (GET and POST request)
- – Read and write JSON

Note

For more information, see “Unit 1- Implement External Data Services” of this course.

Develop a plug-in

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:

Server-side extension points

Allow you to add new services or intercept the request and response messages to mid-tier services layer.

Client-side extension points

Allow you to add new action, new feature, new viewer or completely change the layout of the application.

Required skills

- – Java
- – JavaScript
- – JSON

- HTML5
- CSS3
- Dojo 1.8.4

Note

For more information, see “Unit 2- Develop Plug-ins” and “Unit 3- Create a Custom Workflow Step Processor” of this course.

Integrate custom application

IBM Content Navigator provides two main approaches for integrating custom applications:

Unbound - A lightweight integration with IBM Content Navigator URL API to render a specific desktop, feature, folder, document, or search template in an iframe of your application or in a new page.

Bound - A heavyweight integration, where all the JavaScript code is running directly within your application. Here you can use just JavaScript model and do all the visualization by yourself, or use also the widget library.

To develop a custom application, you can either use IBM Content Navigator GUI and develop your own layout plug-in for your desktop or you can create new application by using IBM Content Navigator JavaScript API.

Note

For more information, see “Unit 4 - Use Content Navigator widgets in other applications” of this course.

1

Implement External Data Services

This unit provides guidance for implementing External Data Services (EDS). The sample EDS is an implementation of the Request and Response filters and it enables IBM Content Navigator to load data dynamically from external sources.

LESSON 1.1: Implement External Data Services

What this lesson is about?

This lesson shows some of the capabilities of the sample External Data Services (EDS) that IBM Content Navigator provides. You can manage property behavior such as prefilled properties, required status, and property validation. Other property behaviors such as adding choice lists and hiding a data field are described in the next lesson.

What you should be able to do?

After completing this lesson, you should be able to use the EDS plug-in for:

- Data validation
- Make a data field as a required field
- Prefill values for the data fields

How you will check your progress?

- Hands on Lab Exercise

Lab Solution files

- The solution files for this lesson are included in the following folder:
  C:\ICN\EDS

References

IBM FileNet P8 5.2 Information Center

IBM DeveloperWorks forums

IBM Developer Works Technical Library

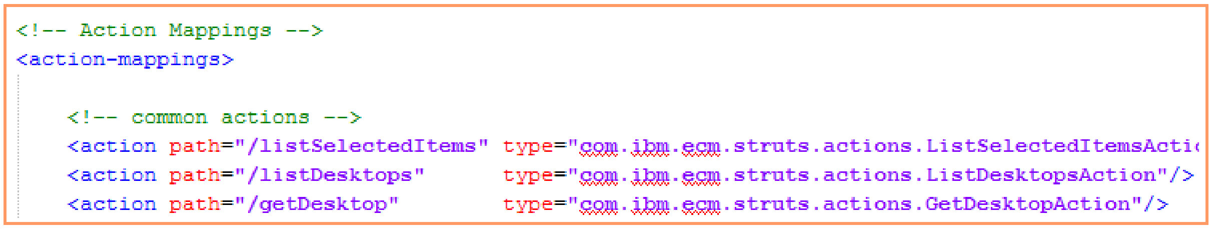
IBM Redbooks publication: Customizing and Extending IBM Content Navigator

Request and Response filters

Request and Response filter overview

- A Request or Response filter stands between the client and the service, and modifies the responses and requests as needed.
- In IBM Content Navigator, when running a service call, a request is sent to the service and the response is sent back from the service to the client tier.
- The IBM Content Navigator framework creates the requests and responses in a standard JSON format.
- The filters enable you to modify:
  - A request before it is sent to the service.
  - A response after it is received from the service.
- A filter is assigned to one service or a list of services within IBM Content Navigator.
  - Provided services are listed in the `struts-config.xml` file, which is in the following directory:

    `<WAS_InstallPath>/profiles/<nameOfProfile>/installedApps/<nameOfCell>/navigator.ear/navigator.war/WEB-INF`
  - The `<action-mappings>` section in the file lists the actions that can be filtered.
  - The `<action path>` tag gives the action name that the `getFilteredServices()` method must return.



```
<!-- Action Mappings -->
<action-mappings>

    <!-- common actions -->
    <action path="/listSelectedItems" type="com.ibm.ecm.struts.actions.ListSelectedItemsActi
    <action path="/listDesktops"      type="com.ibm.ecm.struts.actions.ListDesktopsAction"/>
    <action path="/getDesktop"        type="com.ibm.ecm.struts.actions.GetDesktopAction"/>
```

Request and Response filters and External Data Services (EDS)

- In IBM Content Navigator, the external data service (EDS) is an implementation of the request and response filters.
  - It is an IBM Content Navigator plug-in that allows you to modify property behavior by introducing new REST interface.
  - EDS is designed as a quick and easy starting point for your customization, without the need to implement a plug-in.
- You can also create plug-ins to filter a request that is made to a service, or to filter a response that is received from a service.
  - Example: Check the documents that are included in the request and do an extra level of security check based on external information.

  - Example: Implement extra error handling code when the service is not able to run a specific request.
  - Example: Define special formatting for dedicated columns of a result set through modifying the JSON output by using special Dojo classes.

When to implement Request and Response filters?

You implement request and response filters if you want to do the following type of tasks:

- Reorder properties in the dialog.
- Control data in the action that is not included in EDS.
  - Example: Teamspace Builder pane.
- Add custom control in the document list view when open a folder in Browse view.
  - Example: Set different default sort column for different folder.
- You implement your own request and response filters for a number of tasks.
  - You can use the EDS as an example of the request and response filter implementation.
  - EDS can simplify your implementation.

When to use the sample EDS?

The sample EDS plug-in uses request and response filters to enable a service to provide the EDS functions.

- Use EDS if you want to do the following tasks:
  - Prefill properties with values.
  - Look up the choice list values for a property or dependent property.
  - Set minimum and maximum values.
  - Set property status, such as read-only, required, or hidden.
  - Implement property validation and error checking.

Request and Response filters implemented in the sample EDS

Request filters

The following request filters are included in the sample EDS plug-in:

Filter	Action
AddItemFilter	Add documents and Folders
CheckinFilter	Checkin documents
EditAttributesFilter	Edit Properties
SearchFilter	Search
DispatchItemFilter UpdateStepProcessRequestFilter	Workflow

Response filters

The following response filters are included in the sample EDS plug-in:

Filter	Action
OpenContentClassFilter	Select class
OpenItemFilter	Edit properties
UpdateAttributeDefsFilter	Dependent Properties
OpenSearchTemplateFilter	Search
OpenInbasketFilter OpenProcessorFilter StepProcessVarsFilter	Workflow
UpdateParameterDefsFilter UpdateStepProcessVarsFilter	Dependent parameter in workflow

Actions in IBM Content Navigator that can be implemented with EDS

Based on the implemented request and response filters in EDS, it can be implemented for the following actions in IBM Content Navigator or IBM Content Navigator for Microsoft Office:

- Add documents and folders
- Check in documents to the repository
- Edit properties for an object
- Edit item properties in the viewer

  - Use entry templates
  - Set workflow step properties and workflow filter criteria fields (IBM FileNet P8)
  - Create or use searches
  - Control Content Manager OnDemand search folders

EDS use cases

With EDS, you can manage property behavior such as prefill property values, look up the choice list values, set property status, and validate property values input by the user in real time.

- Provide a choice list for all the estimators for a car insurance claim. This information can be a list of people stored external to the IBM ECM system.
- Provide a dependent choice list of categories that belong to one main category. This option is to limit the available categories under each main category. Example: Regions (main category) with Cities Offices (subcategories).
- Validate entered claim numbers by format (exact number and type of characters).
- Hide input property field.

  Example: Enter a custom text in this field if an insurance claim was opened and the user might not use any of the provided values from the choice list.

External Data Services (EDS)

External Data Service components

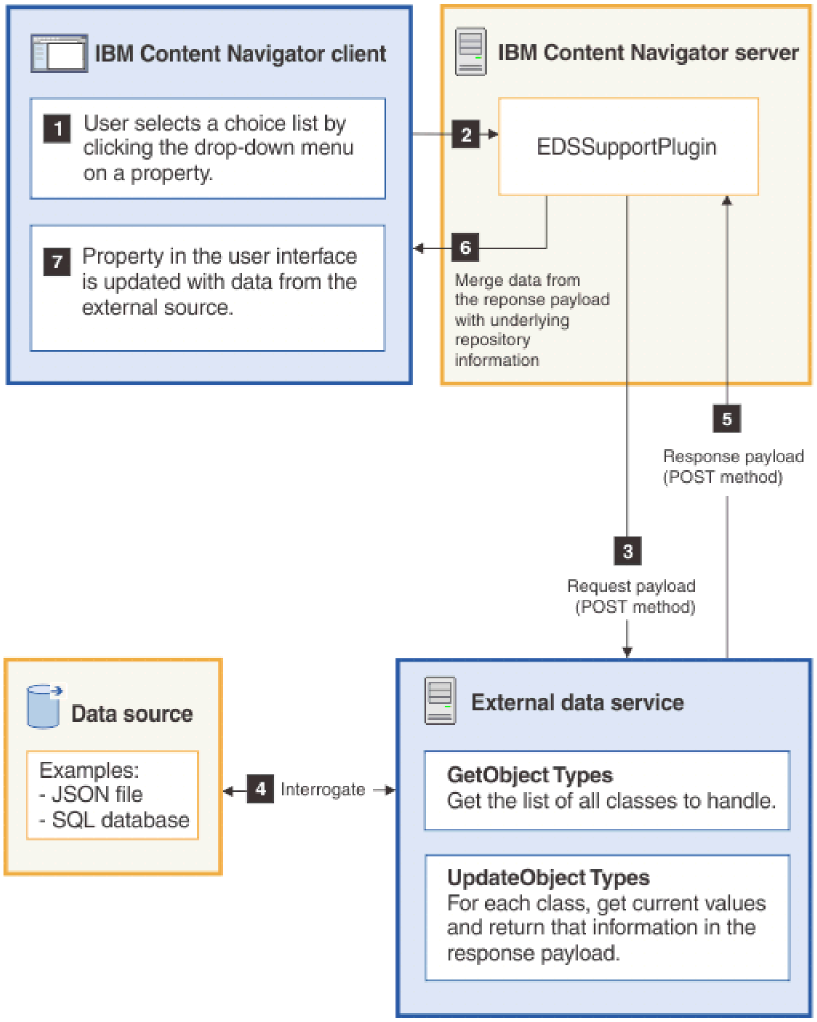
External Data Service Contains two parts:

- EDS plug-in that is used to take EDS function effect.
- EDS web application that is used to get external data and set the property behavior.

EDS architecture diagram

When an EDS is implemented for a certain action or property, the service is started when a business user interacts with that item in the web client.

The following diagram shows that how an external data service submits and returns requests between the IBM Content Navigator client and the EDS.


IBM Content Navigator client
1 User selects a choice list by clicking the drop-down menu on a property.
7 Property in the user interface is updated with data from the external source.
IBM Content Navigator server
2
EDSSupportPlugin
6
Merge data from the reponse payload with underlying repository information
5
Response payload (POST method)
3
Request payload (POST method)
Data source
Examples:
- JSON file
- SQL database
4 Interrogate
External data service
GetObject Types
Get the list of all classes to handle.
UpdateObject Types
For each class, get current values and return that information in the response payload.

EDS Services

The EDS implementation uses two services, which the application developer creates:

- A GetObjectTypes service to get the list of all classes, item types, or workflow information that the external data service needs to handle.
- An UpdateObjectType service for each class to get the current attributes or values and return that information in the response payload.

What happens when EDS is started?

- The EDS is started when a user clicks a data entry field or tries to set a value on a property in the user interface.
- The middle tier service sends a request payload to the EDS, which interrogates the EDS for the requested information about the classes and attributes.
- The information from the response payload is then merged with the underlying information in the repository that you are using, for example, IBM Content Manager or FileNet P8.
- IBM Content Navigator presents the data in the following precedence order:
  - Uses the class attributes
  - Checks for values that the active selected entry template specifies
  - Checks for values that the external data service specifies

IBM Content Navigator sample EDS implementation

You update the sample EDS implementation that is provided with IBM Content Navigator.

- Default directory for the sample EDS (Windows):

  `C:\Program Files(x86)\IBM\ECMClient\samples\sampleEDSService`
- The sample EDS service is a simple web application that contains two Java servlet classes.
  a. **GetObjectTypesServlet**

     Provides an HTTP GET service to determine the list of object types that this EDS implementation supports.

  b. **UpdateObjectTypeServlet**

     Provides the HTTP POST service to obtain external data and dynamically change, validate, and enrich metadata in real time.
- The rest of the files in the sample are JavaScript Object Notation (JSON) files that store the property behavior definition and data list for the EDS implementation.

ObjectTypes.json is the key file that describes which FiletNet P8 Object classes in the IBM ECM repository and the workflow steps that works with the EDS service.

  - The other sample JSON files match some of the Object class names in P8 and contain the sample data that manipulates the property values.

- You must edit the JSON files and configure your system for EDS to work in your system.

EDS REST protocol

You can use the EDS Representational State Transfer (REST) protocol to create an external data service to get data from an external source, such as a file or a table in a database, to customize field properties and manage property behavior in IBM Content Navigator and IBM Content Navigator for Microsoft Office.

- When you create an EDS, your existing data is integrated with IBM Content Navigator field values, but you continue to maintain the data only in the original, authoritative data source.
- You access the data without moving or copying that data to a separate repository, so the source remains in the original data store.
- The EDS must remain available to IBM Content Navigator, so that the external data can be accessed whenever the business user starts the service through the web client.
- You can customize the user interface properties and values without modifying the IBM Content Navigator source code.
  - Upgrades to IBM Content Navigator do not affect the property data that EDS provides.
- IBM Content Navigator EDS plug-in invokes your EDS REST interface implementation without propagation of user’s session and credentials.
  - EDS plug-in sends only the userId in the ClientContext node of the JSON request.

Demonstrations

Register the IBM Content Navigator EDS plug-in

Click here to watch the demonstration.

Set the input field status as "Required" using EDS

Click here to watch the demonstration.

Exercise 1.1.1: Register IBM Content Navigator EDS plug-in

Introduction

The External Data Services (EDS) web application must be deployed before configuring the plug-in. This deployed EDS must be linked to IBM Content Navigator where you register and configure the EDS plug-in (edsPlugin) in the Admin tool.

The sample EDS is already deployed on WebSphere Application Server in your student system. In this lab exercise, you register the EDS plug-in and test it in the IBM Content Navigator.

Procedures

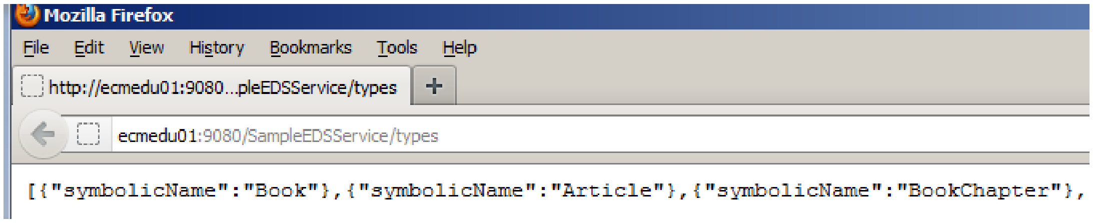
Procedure 1: Check the sample EDS deployment

Procedure 2: Register the EDS plug-in

Procedure 3: Test the plug-in

Procedure 1: Check the sample EDS deployment

1. If it is not already started, start the WebSphere Application Server.
   a. Double-click the Start Server.bat file in the WebSphere Admin folder on the desktop.
   b. Wait for the Start the server page to close.
2. In a Firefox browser, go to `http://ecmedu01:9080/SampleEDSService/types` (The URL is case-sensitive).
3. Verify that the data is shown in a JSON format. On your image, you see more data than the text in the screen capture.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://ecmedu01:9080...pleEDSService/types
ecmedu01:9080/SampleEDSService/types

```
[{"symbolicName":"Book"},{"symbolicName":"Article"},{"symbolicName":"BookChapter"},
```

Procedure 2: Register the EDS plug-in

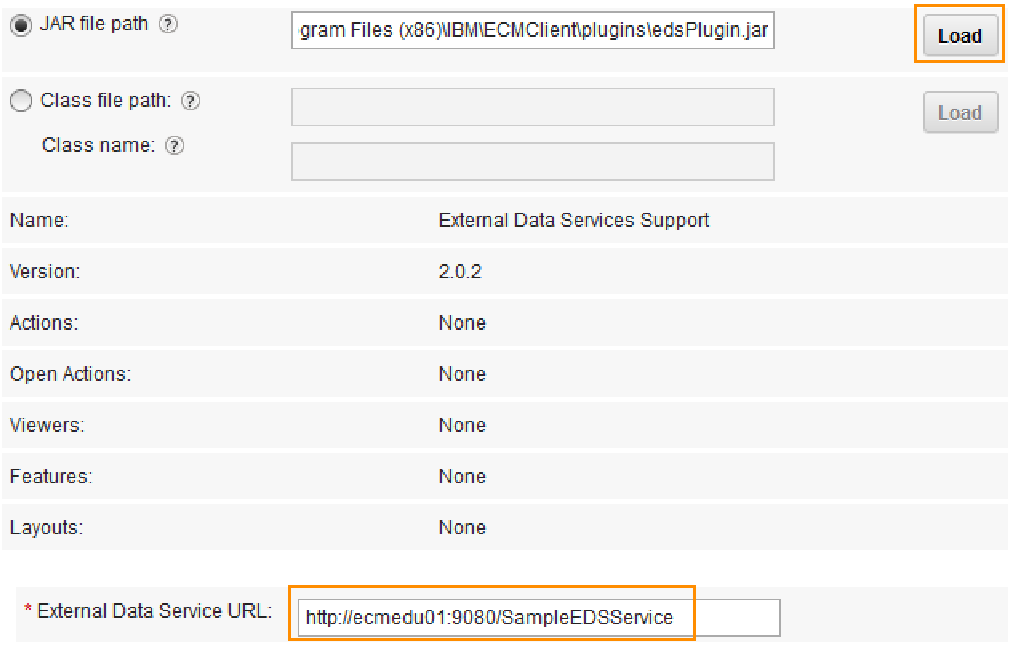
1. In a Firefox browser, start the IBM Content Navigator Administration desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=admin`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In the Admin desktop, click the Plug-ins icon in the left pane.

3. In the Plug-ins tab, click the New Plug-in button.
4. In the New Plug-in page, enter the location for the plug-in JAR file in the File field:
   `C:\Program Files (x86)\IBM\ECMClient\plugins\edsPlugin.jar`

Hint

In Windows Explorer, go to the location of the file. Copy and paste the directory to avoid typing errors.

5. Click Load.
   a. If the file path is valid, the page shows more information as defined for the plug-in.
   b. Enter the value for the External Data Service URL:
   `http://ecmedu01:9080/SampleEDSService`



● JAR file path ? gram Files (x86)\IBM\ECMClient\plugins\edsPlugin.jar

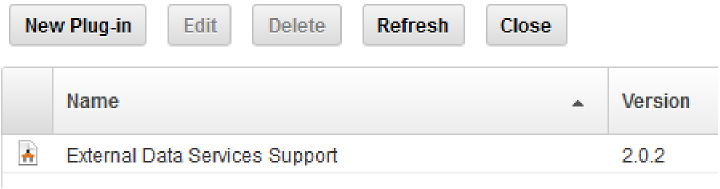
○ Class file path: ?

Class name: ?

Name:	External Data Services Support
Version:	2.0.2
Actions:	None
Open Actions:	None
Viewers:	None
Features:	None
Layouts:	None

* External Data Service URL: http://ecmedu01:9080/SampleEDSService

6. Click Save and Close.
7. Verify that the new plug-in is listed in the Plug-ins tab.



Name	Version
External Data Services Support	2.0.2

8. Log out of IBM Content Navigator and close the browser.

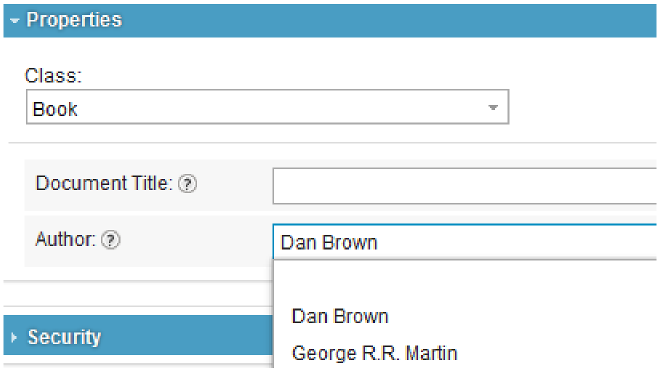
Procedure 3: Test the plug-in

A custom document class (Book class) is created in the Sales repository. EDS contains a JSON file for the Book object class that has a choice list for the properties of this object class.

1. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In the Browse view, if it is not already selected, select the Sales repository.
3. Test the choice list by adding a document.
   a. Open the EDSTest folder and click Add Document.
   b. For "What do you want to save", select "Information about a document" from the list.
   c. For the Properties > Class field, select the `Book` class from the list and click OK.

      In the "Customize a Desktop" unit, if you changed the `Class` label to "`Object Type`", you see the new label.

   d. Click the down-arrow in the Author field and verify that the choice items are shown.



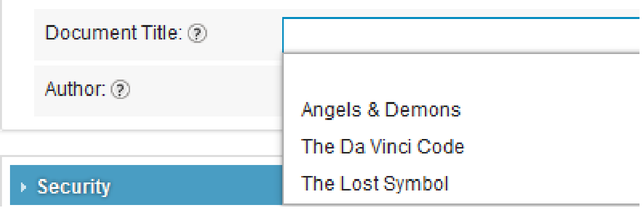
Properties

Class: Book

Document Title:

Author: Dan Brown

4. Check the dependent choice list.
   a. Select a value for the Author field.
   b. Click the down-arrow in the Document Title field.Verify that the values in the list for the Document Title field are dynamically changed to match the Author property value.



Document Title:

Author:

5. Repeat Step 4 with a different author and verify that values are changed.
   a. Click Cancel. Optionally, complete the wizard.
6. Leave the IBM Content Navigator open for the next exercise.

Exercise 1.1.2: Implement property field validation

Introduction

In this exercise, you implement a property field validation in the IBM Content Navigator web client with the sample EDS.

Procedures

Procedure 1: Check the property field validation

Procedure 2: Stop the SampleEDSService and navigator applications

Procedure 3: Edit the JSON file

Procedure 4: Start the SampleEDSService and navigator applications

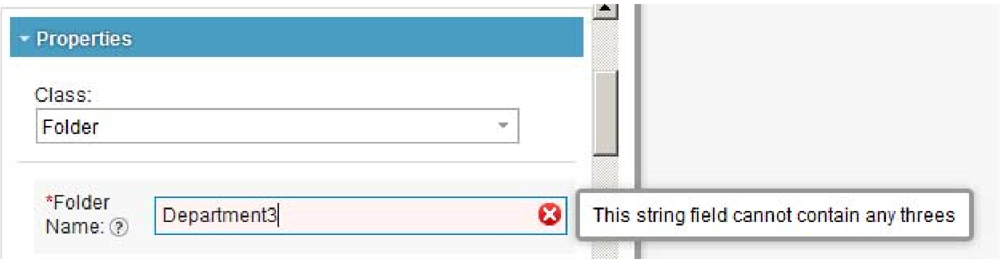
Procedure 5: Test the property field validation

Procedure 1: Check the property field validation

The Folder Name field in the Folder object class uses an existing data validation feature from the sample EDS. The Folder object class is a default class in IBM FileNet P8 repositories.

1. Sample Desktop is already opened. Create a folder.
   a. Select the EDSTest folder.
   b. Click New Folder from the toolbar.
   c. In the Properties section, make sure that Folder is selected for the Class field.

      In the “Customize a Desktop” unit, if you changed the `Class` label to “`Object Type`”, you see the new label.
   d. Enter any value that contains the number 3 for the Folder Name field (Example: `Department3`).
   e. Click Add. Notice that you get an error message:



Class: Folder

*Folder Name: Department3

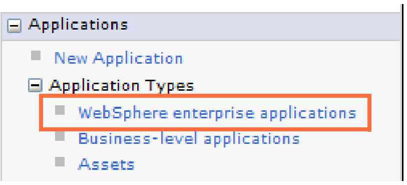
This string field cannot contain any threes

   f. If you remove the number 3, you are able to add the folder.
   g. Log out of IBM Content Navigator and close the browser.

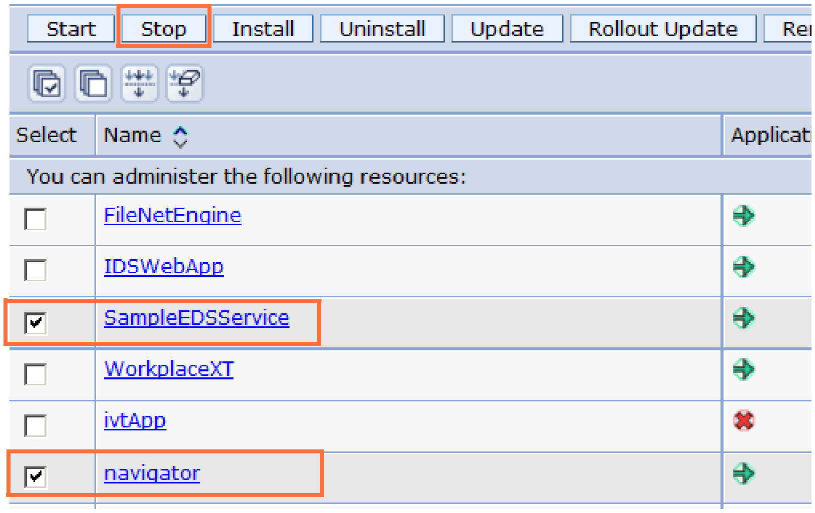
Procedure 2: Stop the SampleEDSService and navigator applications

Stop the navigator and SampleEDSService applications in the WebSphere Application Server administration console, to edit the JSON file. When you start the applications again, the file gets reloaded.

1. In a Firefox browser, click the WAS Admin link in the Bookmarks or go to the following URL: `https://ecmedu01:9043/ibm/console/logon.jsp`
   a. Enter the account information and click *Login*.
      – User ID: p8admin
      – Password: IBMFileNetP8
2. In the left pane, expand Applications > Applications Types.
3. Click the “WebSphere enterprise applications” link.

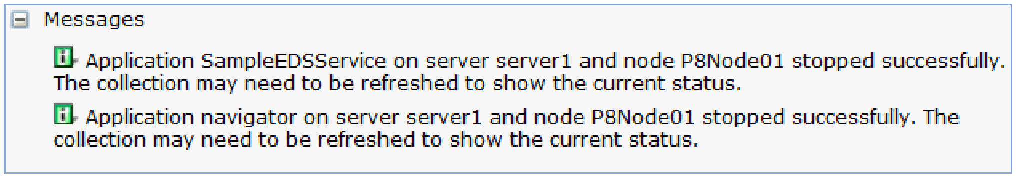

⊟ Applications
New Application
⊟ Application Types
WebSphere enterprise applications
Business-level applications
Assets

4. Select SampleEDSService and navigator in the Enterprise Applications page.



Select	Name	Applicat
You can administer the following resources:		
□	FileNetEngine	➔
□	IDSWebApp	➔
☑	SampleEDSService	➔
□	WorkplaceXT	➔
□	ivtApp	✖
☑	navigator	➔

5. Click Stop and wait for the stop message to display.



⊟ Messages

Application SampleEDSService on server server1 and node P8Node01 stopped successfully. The collection may need to be refreshed to show the current status.

Application navigator on server server1 and node P8Node01 stopped successfully. The collection may need to be refreshed to show the current status.

6. Leave the WebSphere Application Server administration console open.

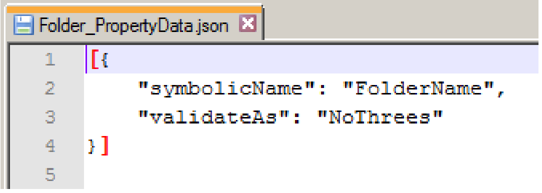
Procedure 3: Edit the JSON file

You edit the corresponding <Classname>_PropertyData.json file to change the sample EDS plug-in behavior for that class. In this procedure, you edit Folder_PropertyData.json so that the Folder Name field does not prompt with an error message when you enter 3. This JSON file is provided in the EDS.

1. Open the file.
   a. In Windows Explorer, go to the EDS service deployment directory:

      ```
      C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\
      P8Node01Cell\sampleEDSService.ear\sampleEDSService.war\WEB-INF\classes
      ```

   b. Open the Folder_PropertyData.json file in a text editor (Notepad++).
   c. If you are prompted for any updates to Notepad++, ignore them for this lab.



```
[{
    "symbolicName": "FolderName",
    "validateAs": "NoThrees"
}]
```

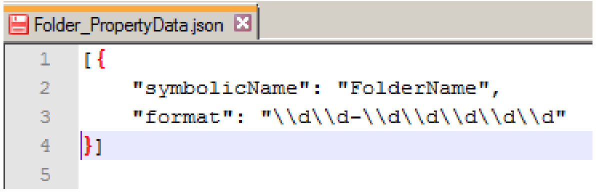
2. Edit the file.
   a. Remove the line `"ValidateAs": "NoThrees"`.

      This step removes the existing validation.

   b. Enter the following line of code to add a different validation:

      ```
      "format": "\\d\\d-\\d\\d\\d\\d\\d"
      ```

   c. The text now looks like the one in the following screen capture.



```
[{
    "symbolicName": "FolderName",
    "format": "\\d\\d-\\d\\d\\d\\d\\d"
}]
```

3. Save and close the file.

Note

The solution `Folder_PropertyData.json` file in the C:\ICN\EDS\Ex.3.1.2 folder for your reference.

Procedure 4: Start the SampleEDSService and navigator applications

Restart the navigator and SampleEDSService applications in the WebSphere Application Server administration console.

1. Select navigator and SampleEDSService in the Enterprise Applications page and click Start. Wait for the Start message to display.
2. Logout of the administration console and close the browser.

Procedure 5: Test the property field validation

1. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Create a folder.
   a. Select the EDSTest folder.
   b. Click New Folder from the toolbar.
   c. In the Properties section, make sure that Folder is selected for the Class field.
   In the “Customize a Desktop” unit, if you changed the `Class` label to “`Object Type`”, you see the new label.
   d. Enter a value for the Folder Name field in this format: `89-45673`

Hint

Because you added another validation, the value must be in a format as shown in the following example: `89-45673.` If the value is not in this format, an error is shown.

   e. Click Add and verify that when you enter a value with `3`, no error is shown.
3. Log out of IBM Content Navigator and close the browser.

Troubleshooting

If the update that you made is not reflected in the Content Navigator web client, continue with the next exercise to troubleshoot. You are going to redeploy the EDS application with updated files in the next exercise.

Exercise 1.1.3: Set the input field status as required

Introduction

In this lab exercise, you set the status of a data input field in the IBM Content Navigator web client as “required” through the sample EDS application. You add a new JSON file and rebuild the application in Eclipse and deploy it in WebSphere Application Server.

Procedures

Procedure 1: View the GetObjectTypesServlet

Procedure 2: Edit the Folder_PropertyData.json file

Procedure 3: Update ObjectTypes.json

Procedure 4: Set the status of a property field as required

Procedure 5: Check the UpdateObjectTypeServlet

Procedure 6: Build a WAR file for the SampleEDSService application

Procedure 7: Deploy the SampleEDSService application

Procedure 8: Verify the sample EDS deployment update

Procedure 9: Reload the EDS plug-in

Procedure 10: Test the EDS Service for the required field configuration

Procedure 1: View the GetObjectTypesServlet

In this procedure, you view the code for GetObjectTypesServlet that provides an HTTP GET service. This service determines the list of object types that the sample EDS implementation supports.

Note

IBM Content Navigator provides a sample External Data Service (EDS) project. This sample EDS project is already imported into the workspace of Eclipse in your student system. Refer to the Appendix: Steps to import the sample EDS into Eclipse of this unit for details about how to set up the project.

1. Open Eclipse by double-clicking the Eclipse_Kepler icon in your desktop.
   a. In the Workspace Launcher page, leave the default workspace directory (`C:\ICN\workspace_Kepler`) and click OK.
2. In Eclipse, expand the sampleEDSService project in the Package Explorer on the left pane.
3. Expand Java Resources > src > (default package).
4. Double-click the `GetObjectTypesServlet.java` file to open.

5. Observe that the following line of code reads the contents of the ObjectTypes.json file.

   ```
   InputStream objectTypesStream = this.getclass().getResourceAsStream
   ("ObjectTypes.json");
   ```

   You update the ObjectTypes.json file to add an object class so that the EDS service responds when the data for this object class is requested.

6. Close the file without any changes.

Procedure 2: Edit the Folder_PropertyData.json file

Note

Do this procedure, if the previous exercise did not work. Otherwise, skip to next procedure.

1. Open the file.
   a. Expand Java Resources > src in the left pane.
   b. Double-click the `Folder_PropertyData.json` file to open.
2. Edit the file.
   a. Remove the line `"ValidateAs": "NoThrees"`.
   b. Enter the following line of code to add a different validation:

   ```
   "format": "\\d\\d-\\d\\d\\d\\d\\d"
   ```

3. Save and close the file.

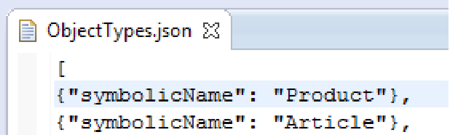
Procedure 3: Update ObjectTypes.json

In this procedure, you update the sample ObjectTypes.json file to include the Product object class. This update in the EDS service instructs the EDS servlet to respond to any request from IBM Content Navigator for data for the Object class with the symbolic name of "Product".

1. In the Package Explorer, expand the Java Resources > src node in the left pane.
   a. Double-click `ObjectTypes.json` to open the file.
2. Edit the file.
   a. Add the following line to the top of the file after the open square bracket symbol `[`.

   ```
   {"symbolicName": "Product"},
   ```

   b. Make sure to add a comma at the end of the line.
   c. The text looks like the following screen capture.



```
[
{"symbolicName": "Product"},
{"symbolicName": "Article"},
```

3. Save and close the file.

Procedure 4: Set the status of a property field as required

In this procedure, you set the product_id property of the Product class as a required field. The required field must have a value to save new documents or edits to the Product class type.

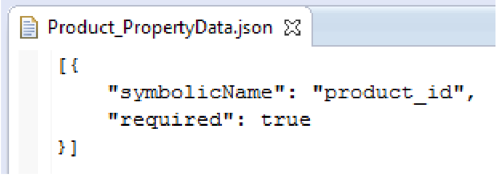
1. Create a file with json file extension.
   a. In the Package Explorer, expand the Java Resources > src node in the left pane.
   b. Right-click src and then select New > Other.
   c. In the New > Select a wizard page, select General > File.
   d. Click Next and enter `Product_PropertyData.json` for the File name field.
      `Product` is the name of the object class for which you want to add data validation.
   e. Click Finish.
2. Add the following text to the file (with the surrounding square bracket`[]` characters).

```
[{
   "symbolicName": "product_id",
   "required": true
}]
```

Hint

Optionally, copy the text from the `C:\ICN\EDS\Ex.3.1.3\Product_PropertyData.json` solution file and paste it into the file.

The text looks like the one in following screen capture.



Product_PropertyData.json

```
[{
    "symbolicName": "product_id",
    "required": true
}]
```

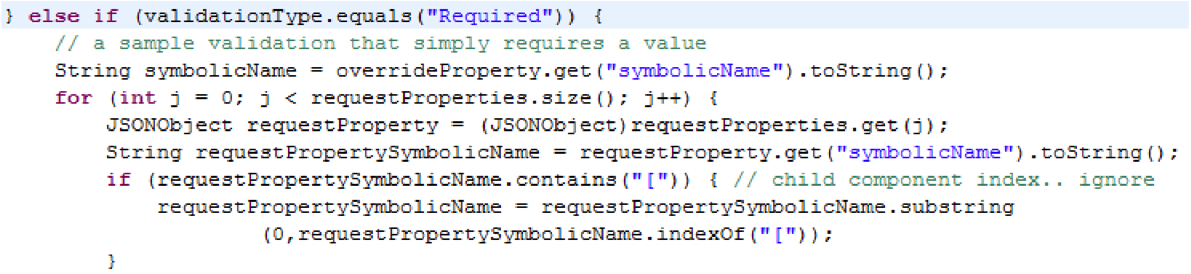
3. Save and close the file.

Procedure 5: Check the UpdateObjectTypeServlet

In this procedure, you check UpdateObjectTypeServlet that provides an HTTP POST service. It gets external data and dynamically changes, validates, and enriches metadata in real time.

1. Expand Java Resources > src > (default package).
   a. Double-click the `UpdateObjectTypeServlet.java` file to open.
2. Observe that the code for the Validation Condition “Required”.
   You provide the key “Required” in the JSON file for an object class so that the EDS service responds when the data for this object class is requested.

3. The screen capture shows the beginning of the block.

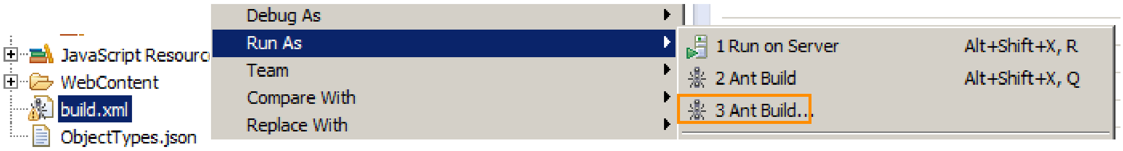


```
} else if (validationType.equals("Required")) {
    // a sample validation that simply requires a value
    String symbolicName = overrideProperty.get("symbolicName").toString();
    for (int j = 0; j < requestProperties.size(); j++) {
        JSONObject requestProperty = (JSONObject)requestProperties.get(j);
        String requestPropertySymbolicName = requestProperty.get("symbolicName").toString();
        if (requestPropertySymbolicName.contains("[")) { // child component index.. ignore
            requestPropertySymbolicName = requestPropertySymbolicName.substring
                    (0,requestPropertySymbolicName.indexOf("["));
        }
```

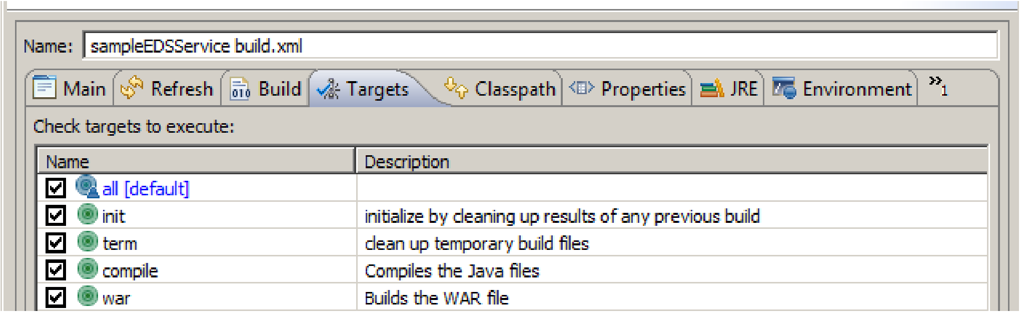
4. Close the file without any changes.

Procedure 6: Build a WAR file for the SampleEDSService application

1. In the Package Explorer, scroll down and right-click build.xml.
2. Select Run As > 3 Ant Build.



3. In the “Edit Configuration” page, select all options and then click Run.

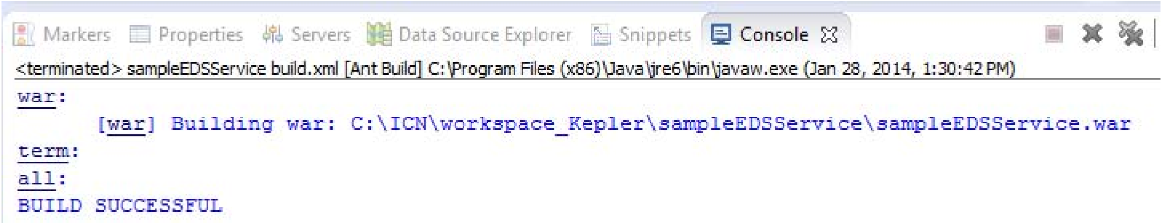


Name: sampleEDSService build.xml

Check targets to execute:

Name	Description
☑ all [default]	
☑ init	initialize by cleaning up results of any previous build
☑ term	clean up temporary build files
☑ compile	Compiles the Java files
☑ war	Builds the WAR file

4. Verify that the WAR file build is successful and check the war file location in the Console tab.



```
<terminated> sampleEDSService build.xml [Ant Build] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (Jan 28, 2014, 1:30:42 PM)
war:
      [war] Building war: C:\ICN\workspace_Kepler\sampleEDSService\sampleEDSService.war
term:
all:
BUILD SUCCESSFUL
```

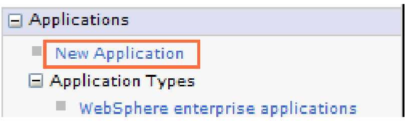
5. Open the Error Log tab.
   a. Click Window > Show View > Other.
   b. In the Show View window, select General > Error Log.

   c. Click OK.
   d. You might see the errors and warnings that are in the following list. Verify that no other errors are shown.
      – Ignore the following warning: `'includeantrurtime' was not set`
      – Because of the automated compilation, you get a message: “`Resource exists on disk`” (repeated multiple times).
6. Leave Eclipse open.

Procedure 7: Deploy the SampleEDSService application

In this procedure, you delete the existing application in the WebSphere Application Server and deploy the new SampleEDSService application that you created.

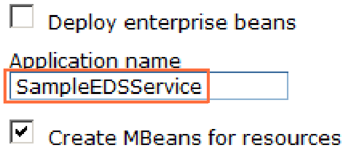
1. In a Firefox browser, log in to the WebSphere Application Server Administration tool.
   a. Click the WAS Admin link in the Bookmarks or go to the following URL: `https://ecmedu01:9043/ibm/console/logon.jsp`
   b. Log in to the WebSphere Application Server Administration tool.
      – User ID: p8admin
      – Password: IBMFileNetP8
2. Stop the application.
   a. In the left pane, expand Applications > Applications Types.
   b. Click the WebSphere enterprise applications link.
   c. Select SampleEDSService in the Enterprise Applications page in the right pane and click Stop.
   d. Wait for the stop message to display.
3. Delete the existing application.
   a. Select SampleEDSService again and click “Uninstall” in the toolbar.
   b. In the “Uninstall Application” page, click OK.
   c. In the Enterprise Applications page, click the Save link.
4. Deploy the application.
   a. In the left pane, expand Applications and click New Application.



5. In the New Application page on the right, click the New Enterprise Application link.
   a. In the “Preparing for the application installation” page, select the Local file system option.
   b. Click Browse and go to the WAR file location: `C:\ICN\workspace_Kepler\sampleEDSService\sampleEDSService.war`
   c. Select the WAR file and click Open.
   d. Click Next.

   e. Leave the default option (Fast Path) and click Next.

6. In the “Select installation options” section, change the Application name to `SampleEDSService` (case sensitive).



□ Deploy enterprise beans

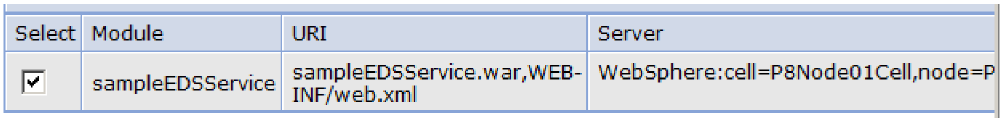
Application name

SampleEDSService

☑ Create MBeans for resources

   a. Scroll down and click Next.

7. In the “Map modules to servers” page, select the check box in the Select column.



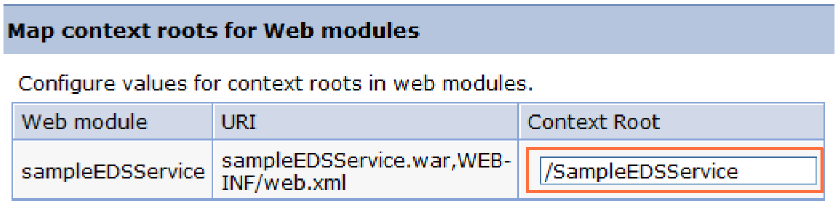
Select	Module	URI	Server
☑	sampleEDSService	sampleEDSService.war,WEB-INF/web.xml	WebSphere:cell=P8Node01Cell,node=P

   a. Click Next.

8. In the “Map virtual hosts for Web modules” page, select the check box in the Select column.

   a. Click Next.

   b. In the “Map context roots for Web modules” page, enter `/SampleEDSService` in the Context Root column.



Map context roots for Web modules

Configure values for context roots in web modules.

Web module	URI	Context Root
sampleEDSService	sampleEDSService.war,WEB-INF/web.xml	/SampleEDSService

   c. Click Next.

9. In the “Metadata for modules” page, select the check box in the metadata-complete attribute column.

   a. Click Next.

10. In the Summary page, scroll down and click Finish.

    It takes a few moments to complete the installation.

11. Save the application.

    a. In the results page, scroll down and click the Save link, to save the changes to the master configuration.

12. Start the application.

    a. In the left pane, expand Applications > Applications Types.

    b. Click the WebSphere enterprise applications link.

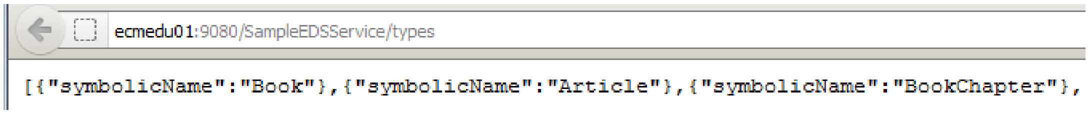
    c. Select SampleEDSService in the Enterprise Applications page in the right pane.

    d. Click Start. Wait for the Start message to display at the top of the page.

13. Logout of the Administration Console and close the browser.

Procedure 8: Verify the sample EDS deployment update

1. In a Firefox browser, go to `http://ecmedu01:9080/SampleEDSService/types` (The URL is case-sensitive).
2. Verify that the data is shown in a JSON format, similar to the one in the screen capture.



```
[{"symbolicName":"Book"},{"symbolicName":"Article"},{"symbolicName":"BookChapter"},
```

3. Copy the application base URL and paste it to a text editor (Notepad) to use it for the next procedure.

Procedure 9: Reload the EDS plug-in

1. In a Firefox browser, start the IBM Content Navigator Administration desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=admin`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In the Admin desktop, click the Plug-ins icon in the left pane.
3. In the Plug-ins tab, click the External Data Services Support plug-in.
4. Click Edit.
5. In the External Data Services Support tab, verify that the “JAR file path” option is selected. If it is not selected, click the option.
6. If the “JAR file path” field is empty, enter the following location for the plug-in JAR file:

   `C:\Program Files (x86)\IBM\ECMClient\plugins\edsPlugin.jar`

Hint

In Windows Explorer, go to the location of the file. Copy and paste the directory to avoid typing errors.

   a. Click Load.
   b. If the External Data Service URL field is empty, enter the following value:

      `http://ecmedu01:9080/SampleEDSService`

   c. Click Save and Close.
   d. Log out of IBM Content Navigator and close the browser.

Procedure 10: Test the EDS Service for the required field configuration

A custom document class (Product) with a property (product_id) is already created in the student system to test the EDS plug-in. In this procedure, you add a document of the Product class in IBM Content Navigator and test the changes.

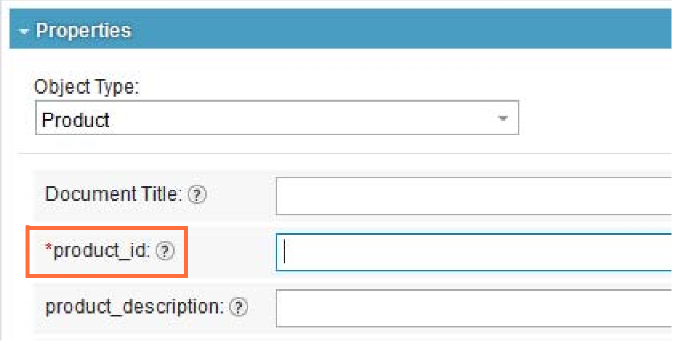
1. Start the IBM Content Navigator Sample Desktop in the Mozilla Firefox browser.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Add a document.
   a. In the Browse view, the Sales repository is already selected. Open the Products folder.
   b. Click Add Document.
   c. For the “What do you want to save” field, select “Information about a document” option.
   d. For the Properties section > Object Type field, select Product from the list and click OK.

   In the “Customize a Desktop” unit, if you changed the `Class` label to “`Object Type`”, you see the new label.
3. Test the EDS: Verify that there is a red star next to the `product_id` property label.

Note

The red star indicates that it is a required field. The Add button is enabled only after entering a value for this required property.

   a. If the property label is not changed, clear the browser cache and test.



Properties

Object Type: Product

Document Title:

*product_id:

product_description:

   b. Enter a value for the `product_id` field (Example: PROD121). The Add button is enabled.
   c. Optionally, complete the Add Document wizard.
4. Logout of Content Navigator and close the Browser.

Exercise 1.1.4: Prefill an initial value for a data field

Introduction

In this lab exercise, you prefill an initial value for a data field in the IBM Content Navigator web client with the sample EDS application.

Procedures

Procedure 1: Stop the navigator and SampleEDSService applications

Procedure 2: Edit the Product_PropertyData.json

Procedure 3: Start the SampleEDSService and navigator applications

Procedure 4: Test the EDS Service for the prefilled value

Procedure 1: Stop the navigator and SampleEDSService applications

Stop the navigator and SampleEDSService applications in the WebSphere Application Server administration console, to edit the JSON file.

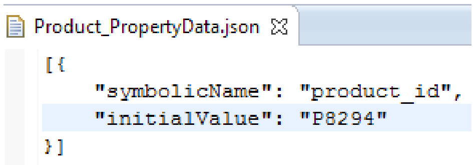
1. In a Firefox browser, click the WAS Admin link in the Bookmarks or go to the following URL: `https://ecmedu01:9043/ibm/console/logon.jsp`
   a. Enter the account information and click *Login*.
      – User ID: p8admin
      – Password: IBMFileNetP8
2. In the left pane, expand Applications > Applications Types.
   a. Click the WebSphere enterprise applications link.
3. Select SampleEDSService and navigator in the Enterprise Applications page.
4. Click Stop and wait for the stop message to display.
5. Leave the WebSphere Application Server administration console open.

Procedure 2: Edit the Product_PropertyData.json

1. In Windows Explorer, go to the EDS service deployment directory:

   ```
   C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\
   P8Node01Cell\sampleEDSService.ear\sampleEDSService.war\WEB-INF\classes
   ```
2. Open the `Product_PropertyData.json` file in a text editor (Notepad++).

3. Edit the file.
   a. Replace the `"required": true` line with the following line:
      `"initialValue": "P8294"`
      `"P8294"` can be any String value.
   b. The text looks like the following screen capture.



```
[{
    "symbolicName": "product_id",
    "initialValue": "P8294"
}]
```

4. Save and close the file.

Note

The solution `Product_PropertyData.json` file in the C:\ICN\EDS\Ex.3.1.4 folder for your reference.

Procedure 3: Start the SampleEDSService and navigator applications

Restart the navigator and SampleEDSService applications in the WebSphere Application Server administration console.

1. Select navigator and SampleEDSService in the Enterprise Applications page and click Start. Wait for the Start message to display.
2. Logout of the administration console and close the browser.

Procedure 4: Test the EDS Service for the prefilled value

In this procedure, you add a document of the Product class in IBM Content Navigator and test the changes.

1. Start the IBM Content Navigator Sample Desktop in Mozilla Firefox browser.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Add a document.
   a. In the Browse view, the Sales repository is already selected.
   b. Open the Products folder.
   c. Click Add Document.
   d. For the “What do you want to save” field, select “Information about a document” option.

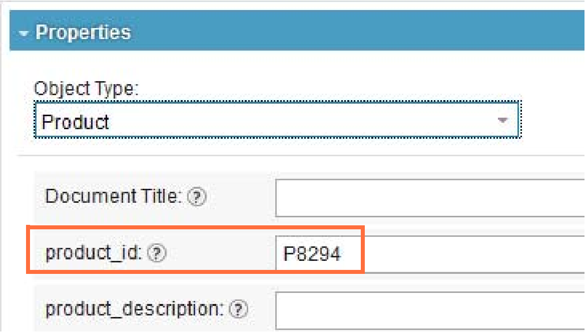
e. For the Properties section > Object Type (which is also referenced as Class) field, select Product from the list.

   In the “Customize a Desktop” unit, if you changed the `Class` label to “`Object Type`”, you see the new label.

f. Click OK.

3. Test the EDS change.

   a. Verify that the value for the `product_id` property field is prefilled.



Properties

Object Type: Product

Document Title:

product_id: P8294

product_description:

   b. Optionally, complete the Add Document wizard.

4. Logout of Content Navigator and close the Browser.

LESSON 1.2: Create Choice Lists with Sample EDS

What this lesson is about?

This lesson shows how to create a choice list and dependent choice lists through External Data Services (EDS).

What you should be able to do?

After completing this lesson, you should be able to use the EDS plug-in for:

- Creating choice lists and dependent choice lists.

How you will check your progress?

- Hands on Lab Exercise

Lab Solution files

- The solution files for this lesson are included in the following folder:
  C:\ICN\EDS

Create choice lists with External Data Service (EDS)

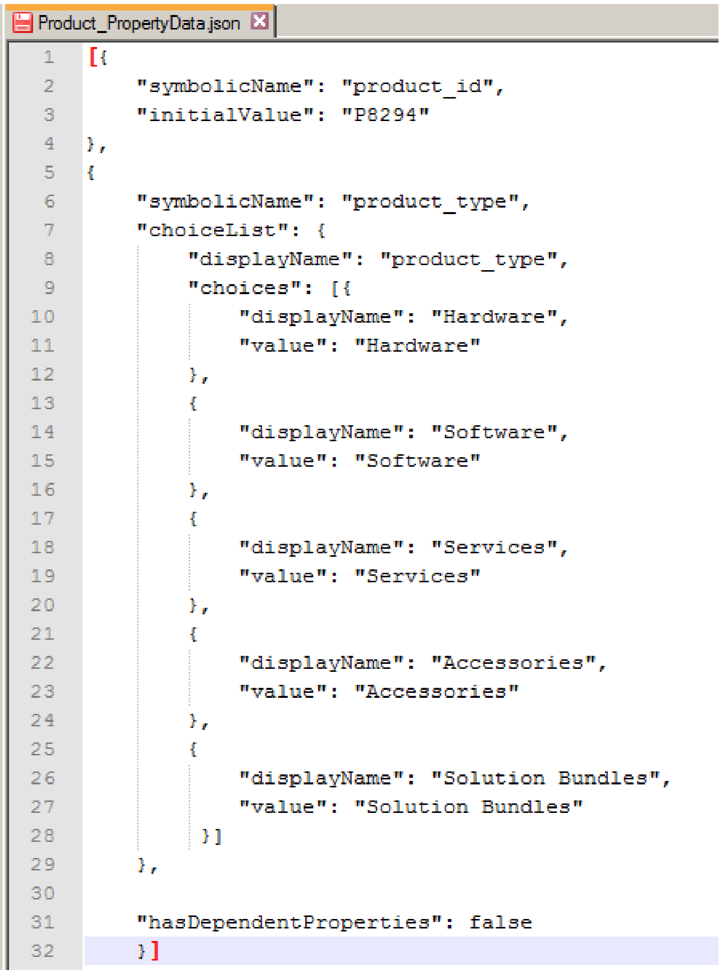
Choice lists

The choice lists are pre-configured so that users avoid any errors from typing the data.

- One of the main features of EDS is to provide values for choice lists. The choice lists can be
  - Hierarchical
  - Dependant

JSON format for the choice lists

The following screen capture shows a sample JSON file.



```
[{
    "symbolicName": "product_id",
    "initialValue": "P8294"
},
{
    "symbolicName": "product_type",
    "choiceList": {
        "displayName": "product_type",
        "choices": [{
            "displayName": "Hardware",
            "value": "Hardware"
        },
        {
            "displayName": "Software",
            "value": "Software"
        },
        {
            "displayName": "Services",
            "value": "Services"
        },
        {
            "displayName": "Accessories",
            "value": "Accessories"
        },
        {
            "displayName": "Solution Bundles",
            "value": "Solution Bundles"
         }]
    },

    "hasDependentProperties": false
    }]
```

- Each object class must have a corresponding JSON file with the same object name.
  - Example: Product_PropertyData.json

- Within each JSON file, square brackets `[]` surround a list of object properties to represent an array.
- Within the array, curly braces `{}` surround each property with its attributes.
- Within each property, you can have arrays of attributes for that property such as choice items.
  - The line `"symbolicName": "product_id"` denotes the ID of the property as defined in the repository.
  - After this line, include any instructions for the EDS for that property.

    Example: `"required": true`
- You can include choice list as the instruction for a property: `"choiceList": {`
  - Provide a display name for the choice list: `"displayName": "product_type",`
  - Provide an array of choice values in the following format:

```
"choices": [{
      "displayName": "Hardware",
      "value": "Hardware"
   },
   ...
   {
      "displayName": "Solution Bundles",
      "value": "Solution Bundles"
}]
```

  - The last line `"hasDependentProperties": false` indicates that no other properties depend on the values in this choice list.

Dependent choice lists

- The values of one choice list depend on the selection of a value for another property.
- The following line of code indicates that other property values change depending on the value of this property: `"hasDependentProperties": true`

Exercise 1.2.1: Add an external choice list with EDS

Introduction

You can add a choice list to a property of an object class for the IBM FileNet P8 repositories in Administration Console for Content Platform Engine (internal choice lists).

In this lab exercise, you add an external choice list for the product_type property of the Product class with EDS.

Procedures

Procedure 1: Stop the navigator and SampleEDSService applications

Procedure 2: Add an external choice list

Procedure 3: Start the SampleEDSService and navigator applications

Procedure 4: Test the EDS Service for the choice list

Procedure 1: Stop the SampleEDSService and navigator applications

Stop the navigator and SampleEDSService applications in the WebSphere Application Server administration console, to edit the JSON file.

1. In a Firefox browser, click the WAS Admin link in the Bookmarks or go to the following URL: `https://ecmedu01:9043/ibm/console/logon.jsp`
   a. Enter the account information and click *Login*.
      – User ID: p8admin
      – Password: IBMFileNetP8
2. In the left pane, expand Applications > Applications Types.
3. Click the WebSphere enterprise applications link.
4. Select SampleEDSService and navigator in the Enterprise Applications page.
5. Click Stop and wait for the stop message to display.
6. Leave the WebSphere Application Server administration console open.

Procedure 2: Add an external choice list

1. Open the file.
   a. In Windows Explorer, go to the EDS service deployment directory:

   ```
   C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\
   P8Node01Cell\sampleEDSService.ear\sampleEDSService.war\WEB-INF\classes
   ```

   b. Open the `Product_PropertyData.json` file in a text editor (Notepad++).

2. Edit the file.
    a. After the curly braces `}` bracket in the last line (before the square bracket symbol `]`), add a comma and press Enter to insert a new line.
    b. Add the following code to the file.

Note

Optionally, copy the text from the `C:\ICN\EDS\Ex.3.2.1\Product_PropertyData.json` file and paste the code into the file.

```
{
    "symbolicName": "product_type",
    "choiceList": {
        "displayName": "product_type",
        "choices": [{
            "displayName": "Hardware",
            "value": "Hardware"
        },{
            "displayName": "Software",
            "value": "Software"
            },{
            "displayName": "Services",
            "value": "Services"
        },{
            "displayName": "Accessories",
            "value": "Accessories"
        },{
            "displayName": "Solution Bundles",
            "value": "Solution Bundles"
        }]
    },
    "hasDependentProperties": false
}
```

Note

The last line `"hasDependentProperties": false` indicates that no other properties depend on the values in this choice list.

3. Save and close the file.

Procedure 3: Start the SampleEDSService and navigator applications

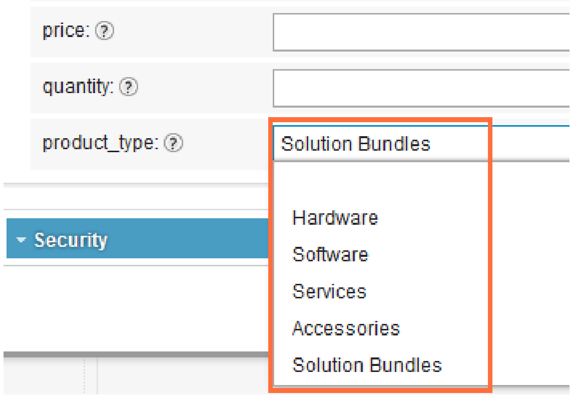
Restart the applications in the WebSphere Application Server administration console.

1. Select navigator and SampleEDSService in the Enterprise Applications page and click Start. Wait for the Start message to display.
2. Logout of the administration console and close the browser.

Procedure 4: Test the EDS Service for the choice list

A custom class (Product) with a property (product_type) is created in the student system. In this procedure, you add a document (Product class) in IBM Content Navigator and test the changes.

1. Start the IBM Content Navigator Sample Desktop in Mozilla Firefox browser.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Add a document.
   a. In the Browse view, the Sales repository is already selected.
   b. Open the Products folder and click Add Document.
   c. For the “What do you want to save” field, select “Information about a document” option.
   d. In the Class field, select Product from the list and click OK.
   In the “Customize a Desktop” unit, if you changed the `Class` label to “`Object Type`”, you see the new label.
3. Test the EDS change.
   a. Click the `product_type` property.
   b. Verify that the property has a choice list that is associated with it. You are able to select a value from the choices that you entered.



4. Optionally, complete the Add Document wizard.
5. Logout of the Content Navigator and close the browser.

Exercise 1.2.2: Create dependent choice lists with EDS

Introduction

Recall the Book object class example from the previous lesson (Exercise 3.1.1). When you select an author name from the Author choice list, the choice list value for the Document Title changes automatically to match the author value. The Author and the Document Title choice lists represent dependent choice lists.

In this exercise, the dependent choice lists are in the Region and BranchOffice properties. When the user selects a Region, the BranchOffice choice list is updated to reflect choices that are valid for a certain region.

Procedures

Procedure 1: Create dependent choice lists

Procedure 2: Test the EDS Service for the dependent choice list

Procedure 1: Create dependent choice lists

The Invoice_PropertyData.json file is available in the sample EDS application, so no changes are required. In this procedure, you check the code.

1. If it is not already open, open Eclipse by double-clicking the Eclipse_Kepler icon in your desktop.
   a. In the Workspace Launcher page, leave the default workspace directory (`C:\ICN\workspace_Kepler`) and click OK.
2. In Eclipse, expand the sampleEDSService project in the Package Explorer on the left pane.
   a. Expand Java Resources > src and double-click the `Invoice_PropertyData.json` file to open.
3. Check the `Region` property.
   a. Observe that the `Region` property block, contains a choice list similar to the one you added in the previous lab exercise.
   b. Verify the following line of code at the end of the block (before the curly braces).

   ```
   "hasDependentProperties": true
   ```

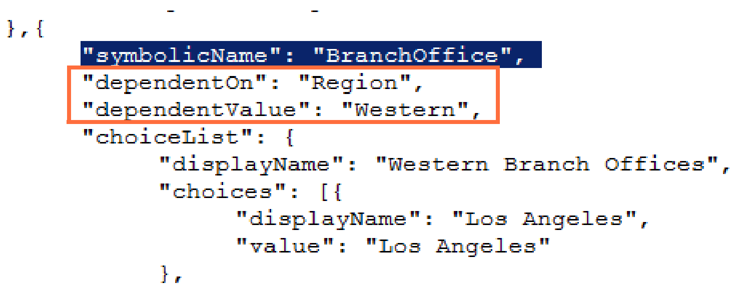
   This line indicates that other property values change depending on the value of this property.
4. Check the `BranchOffice` property in the next block.
   a. Observe that the `BranchOffice` property block also contains a choice list.
   b. The following line of code indicates that the `BranchOffice` property value depends on the value of `Region` property.

   ```
   "dependentOn": "Region",
   ```

c. Since the `Region` property is a choice list that contains many possible values, the following line of code specifies a particular value within that choice list.

   `"dependentValue": "Western",`

   Observe that there are many blocks of `BranchOffice` property in this file, one for each value in the `Region` choice list. When a user selects a value for Region, the choice items for BranchOffice change as defined in this JSON file.



```
},{
          "symbolicName": "BranchOffice",
          "dependentOn": "Region",
          "dependentValue": "Western",
          "choiceList": {
                    "displayName": "Western Branch Offices",
                    "choices": [{
                              "displayName": "Los Angeles",
                              "value": "Los Angeles"
                    },
```

5. Close the file without any changes.

Procedure 2: Test the EDS Service for the dependent choice list

A custom document class (Invoice) with required properties for this exercise is already created in the student system. In this procedure, you add a document (Invoice class) in IBM Content Navigator and test the changes to the EDS plug-in.

1. Start the IBM Content Navigator Sample Desktop in Mozilla Firefox browser.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`.
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Add a document.
   a. In the Browse view, the Sales repository is already selected.
   b. Open the `EDSTest` folder and click Add Document.
   c. For the “What do you want to save” field, select “Information about a document” option.
   d. In the Class field, select `Invoice` from the list and click OK.
      In the “Customize a Desktop” unit, if you changed the `Class` label to “`Object Type`”, you see the new label.
3. Test the EDS change.
   a. Select `Western` from the list as the value for the `Region` property.
   b. Verify that the choice list for the Branch Office matches the Region value.
   c. Select another value for the Region and verify that the Branch Office values changes.
   d. Optionally, complete the Add Document wizard.
   e. Log out of the IBM Content Navigator and close the Browser.

Exercise 1.2.3: Change the field status dynamically for a property

Introduction

You can dynamically change the field status for a property by editing the corresponding JSON file in the EDS sample application.

In this example, you change the field status for the `quantity` property of the Product class to “hidden” or “not hidden” depending on the value of the `product_type` property (value is selected from a choice list) of the Product class.

If the value of the `product_type` property is “Service” or no value is selected for this property, then the `quantity` property is hidden from the display when you add a document or edit properties in the Browse or Search views. For all other values, the `quantity` property is shown.

Procedures

Procedure 1: Stop the navigator and SampleEDSService applications

Procedure 2: Edit the Product_PropertyData.json

Procedure 3: Start the SampleEDSService and navigator applications

Procedure 4: Test the EDS Service for the field status

Procedure 1: Stop the navigator and SampleEDSService applications

Stop the navigator and SampleEDSService applications in the WebSphere Application Server administration console, to edit the JSON file.

1. In a Firefox browser, click the WAS Admin link in the Bookmarks or go to the following URL: `https://ecmedu01:9043/ibm/console/logon.jsp`
   a. Enter the account information and click *Login*.
      - User ID: p8admin
      - Password: IBMFileNetP8
2. In the left pane, expand Applications > Applications Types.
   a. Click the WebSphere enterprise applications link.
3. Select SampleEDSService and navigator in the Enterprise Applications page.
4. Click Stop and wait for the stop message to display.
5. Leave the WebSphere Application Server administration console open.

Procedure 2: Edit the Product_PropertyData.json

1. In Windows Explorer, go to the EDS service deployment directory:

   ```
   C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\
   P8Node01Cell\sampleEDSService.ear\sampleEDSService.war\WEB-INF\classes
   ```

2. Replace the `Product_PropertyData.json` file with the solution file.

   a. Delete the `Product_PropertyData.json` file.

   b. Copy the solution `Product_PropertyData.json` file from the C:\ICN\EDS\Ex.3.2.3 folder. Paste it in the EDS service deployment directory that is specified in Step 1.

3. Open the `Product_PropertyData.json` file in a text editor (Notepad++).
4. Observe that the `product_type` property block contains a choice list that you added in the previous lab exercise.
5. The line of code at the end of the block indicates that other property values change depending on the value of this property.

   ```
   "hasDependentProperties": true
   ```

6. Observe the quantity property block.

   a. In the following lines of code, the "`dependentOn`" line indicates that the `quantity` property behavior depends on the value of `product_type` property.

   b. Since the `product_type` property is a choice list that contains many possible values, the "`dependentValue`" line specifies a particular value within that choice list.

   c. The next line decides whether to hide the display of `quantity` property.

   ```
   {
      "symbolicName": "quantity",
      "dependentOn": "product_type",
      "dependentValue": "Hardware",
      "hidden": false
   },
   ```

   d. Observe that there are many blocks of `quantity` property in this file. In each block, the "dependentValue" line has a choice value from the product_type choice list. When a user selects a value for product_type, the `quantity` property behavior changes as defined in this JSON file.

7. Close the file without any changes.
8. Exit Eclipse by clicking File > Exit.

Procedure 3: Start the SampleEDSService and navigator applications

Restart the applications in the WebSphere Application Server administration console.

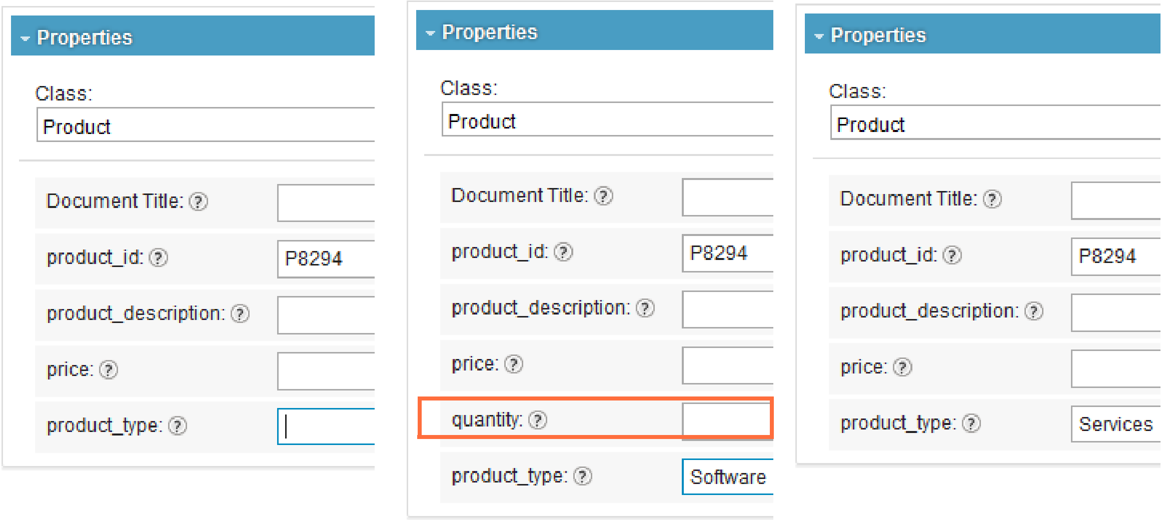
1. Select navigator and SampleEDSService in the Enterprise Applications page and click Start. Wait for the Start message to display.
2. Logout of the administration console and close the browser.

Procedure 4: Test the EDS Service for the field status

A custom document class (Product) with required properties for this exercise is already created in the student system. In this procedure, you add a document (Product class) in IBM Content Navigator and test the changes to the EDS plug-in.

1. Start the IBM Content Navigator Sample Desktop in Mozilla Firefox browser.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Add a document.
   a. In the Browse view, the Sales repository is already selected.
   b. Open the `EDSTest` folder and click Add Document.
   c. For the “What do you want to save” field, select “Information about a document” option.
   d. In the Class field, select `Product` from the list and click OK.

   In the “Customize a Desktop” unit, if you changed the `Class` label to “`Object Type`”, you see the new label.
3. Test the EDS change.
   a. Notice that when no value is selected for the `product_type` field, `quantity` field is not visible.
   b. Select `Software` from the list as the value for the `product_type` property.
   c. Verify that the `quantity` field is visible.
   d. Select `Services` as the value for the `product_type` property.
   e. Verify that the `quantity` field is hidden.



Properties
Class: Product
Document Title:
product_id: P8294
product_description:
price:
product_type:

Properties
Class: Product
Document Title:
product_id: P8294
product_description:
price:
quantity:
product_type: Software

Properties
Class: Product
Document Title:
product_id: P8294
product_description:
price:
product_type: Services

   f. Optionally, complete the Add Document wizard.
   g. Log out of the IBM Content Navigator and close the Browser.

Additional Information

Implement EDS on workflow properties

You can implement EDS on a workflow for the IBM FileNet P8 system.

- You can manage the property behavior for the workflow step data fields.
- The JSON files must be modified to match your IBM Content Navigator workflow.

Note

Refer to the “Create a Custom Workflow Step Processor” unit in this course for the details about implementing EDS on a workflow.

Update the sample EDS to use an external database

You can update the sample EDS to use an external database instead of JSON data files.

Note

Refer to the Redbook “SG248055 - Customizing and Extending Content Navigator” to create a service that returns data programatically, through calls to a relational database or 3rdParty system. The DDL files that are used in the Redbook are included in the C:\ICN\EDS_Database folder.

Programatically creating custom request and response filters

You can create custom request and response filters programmatically (as IBM Content Navigator plug-ins) if your business requires a task that is not included in the EDS implementation.

Note

Refer to the “Develop Plug-ins” unit in this course for details about implementing EDS on a workflow.

Appendix: Steps to import the sample EDS into Eclipse

Introduction

IBM Content Navigator provides a sample External Data Service (EDS) project. In this appendix, you import the sample EDS project into the workspace of Eclipse.

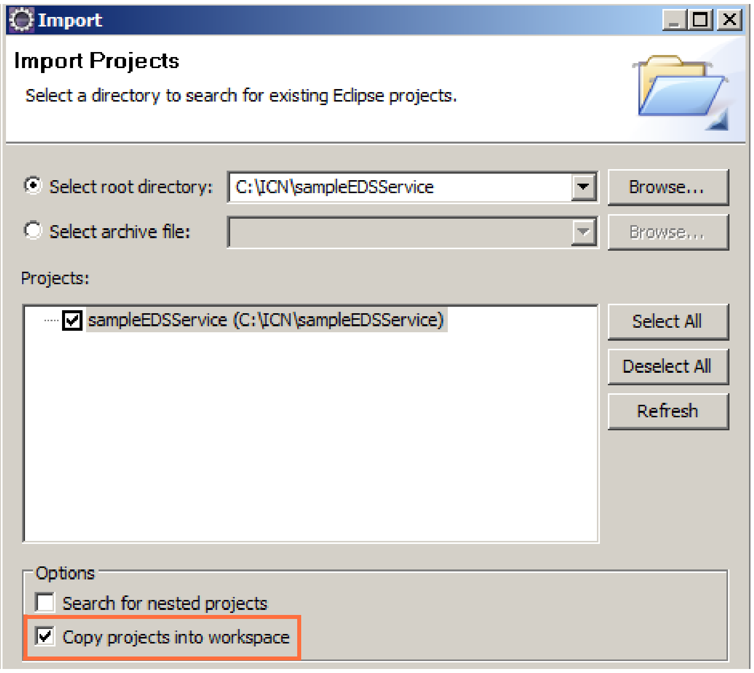
Procedure 1: Import the sample EDS project into Eclipse

1. Open Eclipse by double-clicking the Eclipse_Kepler icon in your desktop.
2. In the Workspace Launcher page, leave the default workspace directory (`C:\ICN\workspace_Kepler`) and click OK.
3. Click File > Import, select "Existing project into workspace" as the option.
   a. Click Next.
4. In the Import Projects page, enter the path to the provided sampleEDSService directory.

Note

The default directory for the sample EDS (Windows) is `C:\Program Files(x86)\IBM\ECMClient\samples\sampleEDSService`. A copy of this folder is created in the `C:\ICN\sampleEDSService` directory.

   a. Select the “Copy projects into workspace” option.



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\ICN\sampleEDSService

○ Select archive file:

Projects:

☑ sampleEDSService (C:\ICN\sampleEDSService)

Options

□ Search for nested projects

☑ Copy projects into workspace

   b. Click Finish.

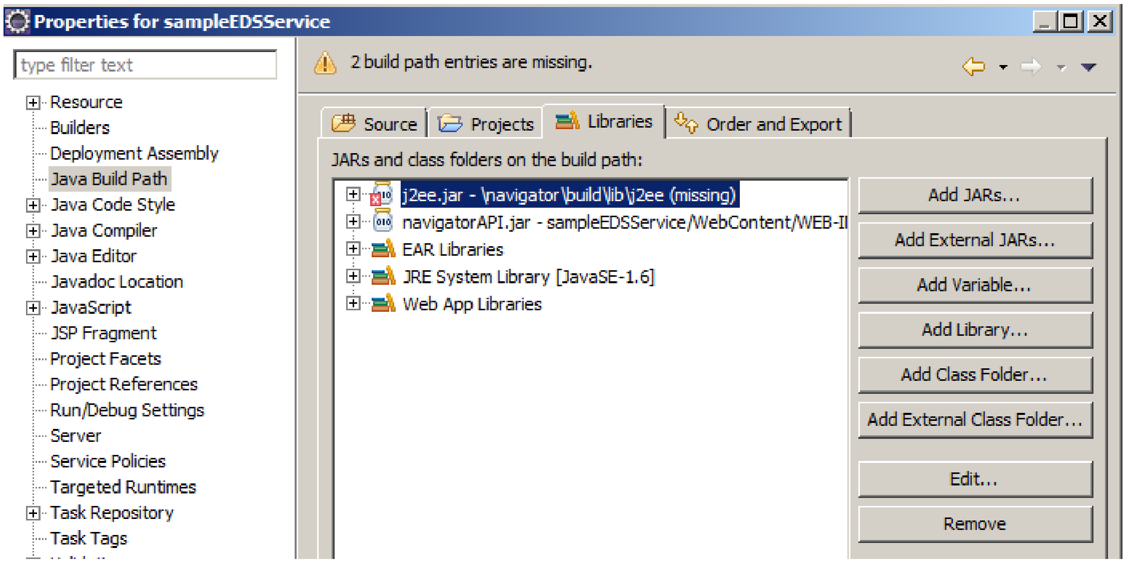
Appendix: Steps to import the sample EDS into Eclipse

(Continued)

Note

After importing the existing project, a build path problem occurs with the configured reference to the j2ee.jar file. Fix this error in the following steps.

5. Right-click the sampleEDSService project and click Properties.
   a. In the Properties for sampleEDSService page, select Java Build Path from the left pane.
6. Click Libraries tab and select j2ee.jar.

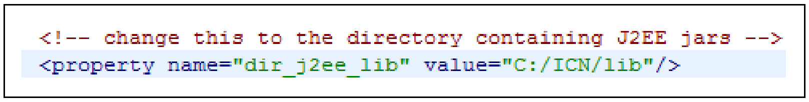


2 build path entries are missing.

JARs and class folders on the build path:

- j2ee.jar - \navigator\build\lib\j2ee (missing)
- navigatorAPI.jar - sampleEDSService/WebContent/WEB-II
- EAR Libraries
- JRE System Library [JavaSE-1.6]
- Web App Libraries

   a. Click Edit.
   b. Go to the C:\ICN\lib folder and select the j2ee.jar file.
   c. Click Open.
7. Click the Projects tab and select navigator (missing).
   a. Click Remove.
   b. In the Properties for sampleEDSService page, click OK.
8. Modify the `build.xml` file that is used to generate a WAR file.
   a. Edit the “`dir_j2ee_lib`” property value to point to the j2ee directory on your system.



```
<!-- change this to the directory containing J2EE jars -->
<property name="dir_j2ee_lib" value="C:/ICN/lib"/>
```

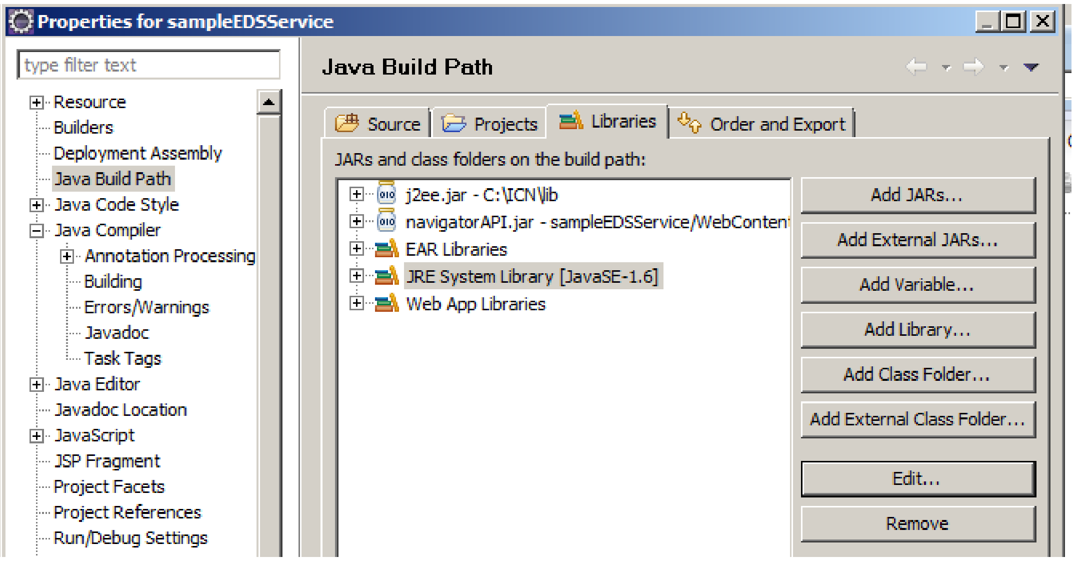
9. Save and close the file.

Appendix: Steps to import the sample EDS into Eclipse

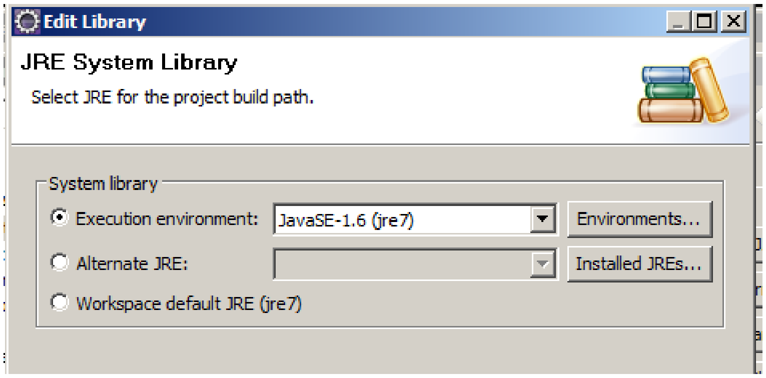
(Continued)

Procedure 2: Verify that the correct JRE is referenced for the project

1. In the Properties for sampleEDSService page, select Java Build path in the left pane.
   a. Click the Libraries tab in the right pane.
2. Select JRE System Library [JavaSE-1.6] and click Edit.



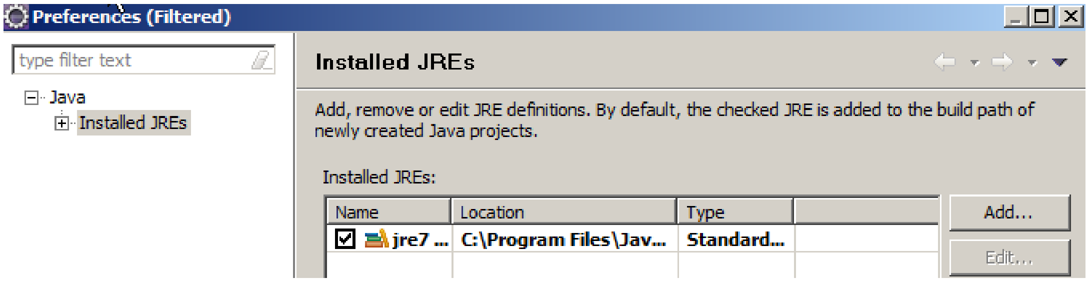
3. In the JRE System Library page, click Installed JREs.



● Execution environment: JavaSE-1.6 (jre7)

Appendix: Steps to import the sample EDS into Eclipse

4. In the Preferences > Installed JREs page, click Add.

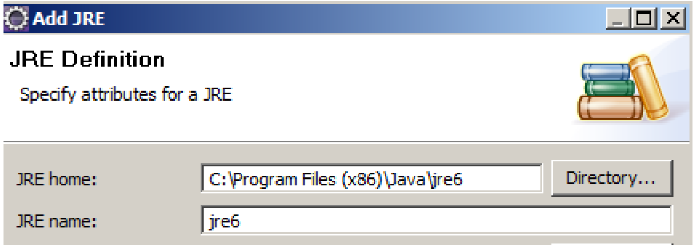


Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

Name	Location	Type	
☑ jre7 ...	C:\Program Files\Jav...	Standard...	

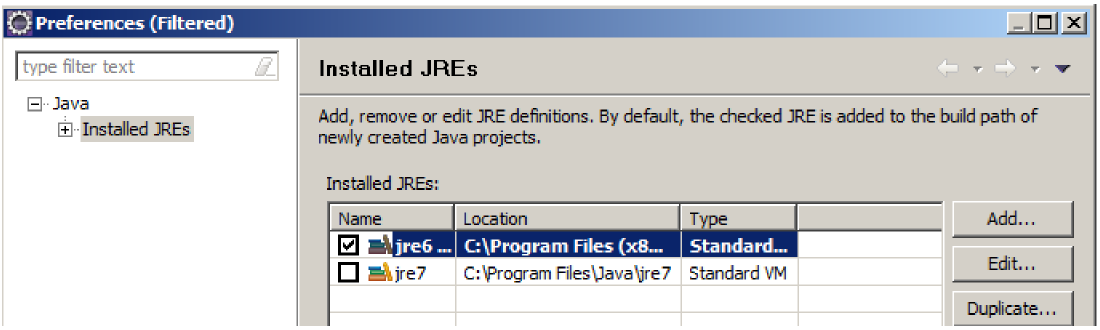
5. In the JRE type page, click Standard VM and click Next.
6. In the Add JRE > JRE Definition page, click Directory for the JRE home field.
7. Go to C:\Program Files (X86)\Java\jre6.



JRE home: C:\Program Files (x86)\Java\jre6

JRE name: jre6

8. Click Finish.
9. In the Preferences > Installed JREs page, verify that jre6 is added and selected.



Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

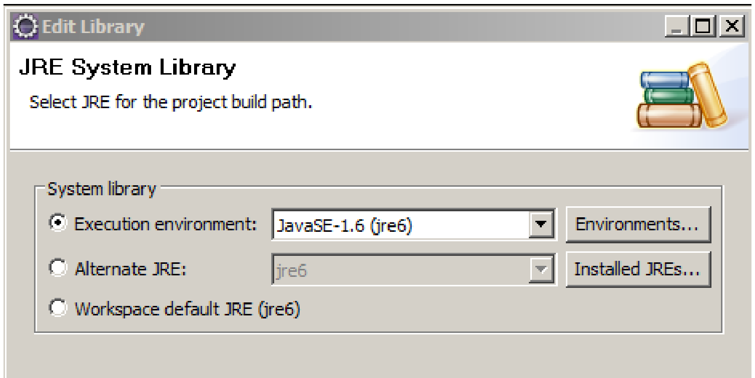
Installed JREs:

Name	Location	Type	
☑ jre6 ...	C:\Program Files (x8...	Standard...	
☐ jre7	C:\Program Files\Java\jre7	Standard VM	

10. Click OK.

Appendix: Steps to import the sample EDS into Eclipse

11. Back in the JRE System Library page, click Finish.



Edit Library

JRE System Library

Select JRE for the project build path.

System library

● Execution environment: JavaSE-1.6 (jre6)

○ Alternate JRE: jre6

○ Workspace default JRE (jre6)

12. Back in the Properties for sampleEDSService page, click OK.
13. Exit Eclipse.

2

Develop Plug-ins

This unit provides guidance for customizing IBM Content Navigator with the custom plug-ins.

LESSON 2.1: IBM Content Navigator Development Overview

What this lesson is about

This lesson gives an overview of the IBM Content Navigator development architecture, visual widget, and modeling libraries and the plug-ins.

What you should be able to do

After completing this lesson, you should be able to:

- Identify the development components of IBM Content Navigator.
- Set up the Eclipse development environment.

How you will check your progress?

- Hands of labs.

References

IBM FileNet P8 5.2 Information Center

IBM Content Navigator JavaScript API reference

IBM Content Navigator Java API reference

IBM DeveloperWorks forums

IBM Developer Works Technical Library

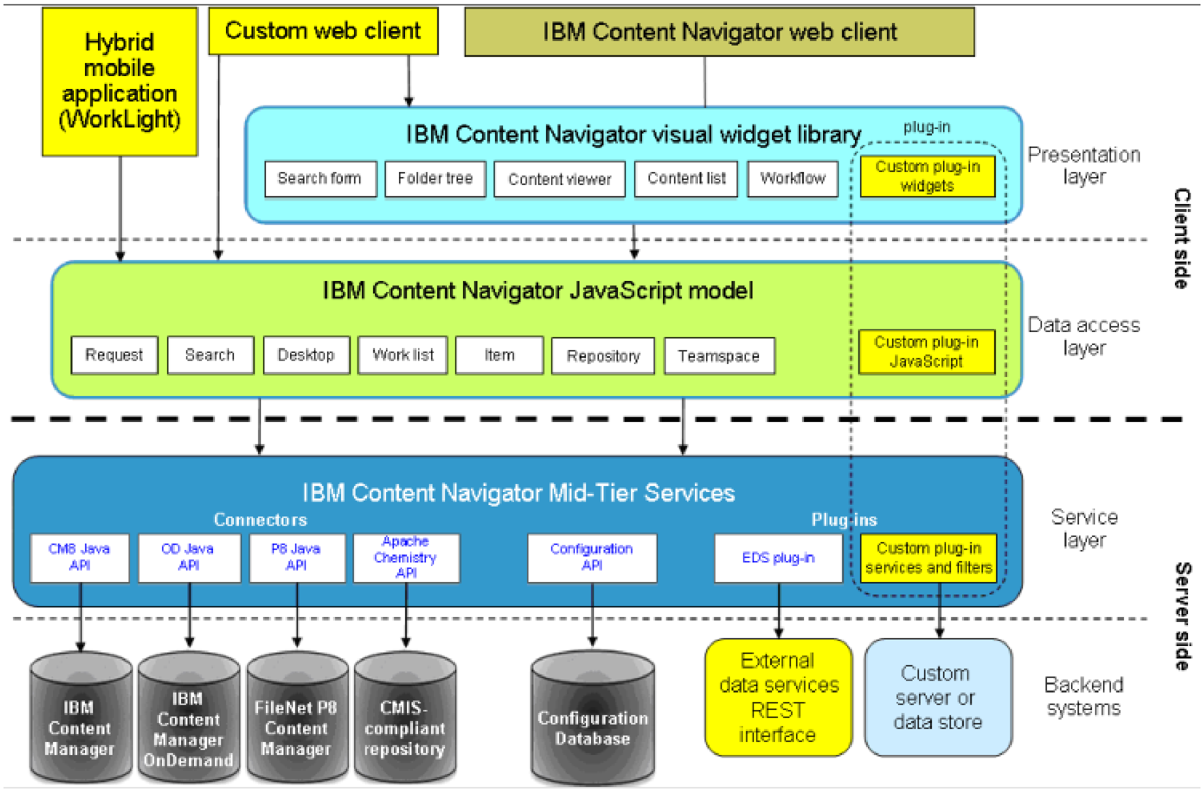
IBM Redbooks publication: Customizing and Extending IBM Content Navigator

IBM Content Navigator development architecture

Architectural diagram

IBM Content Navigator is a framework that provides tools and interfaces to build customized applications for your business needs.

It uses a Model-View-Controller (MVC) architecture. The components are separated functionally according to the MVC design as shown in the following diagram.


Hybrid mobile application (WorkLight)
Custom web client
IBM Content Navigator web client
IBM Content Navigator visual widget library
plug-in
Search form
Folder tree
Content viewer
Content list
Workflow
Custom plug-in widgets
Presentation layer
Client side
IBM Content Navigator JavaScript model
Request
Search
Desktop
Work list
Item
Repository
Teamspace
Custom plug-in JavaScript
Data access layer
IBM Content Navigator Mid-Tier Services
Connectors
Plug-ins
CM8 Java API
OD Java API
P8 Java API
Apache Chemistry API
Configuration API
EDS plug-in
Custom plug-in services and filters
Service layer
Server side
IBM Content Manager
IBM Content Manager OnDemand
FileNet P8 Content Manager
CMIS-compliant repository
Configuration Database
External data services REST interface
Custom server or data store
Backend systems

Note

The yellow boxes in the diagram highlight the parts where you can implement your own code to create new or to customize the existing application.

The diagram shows several layers of the IBM Content Navigator framework.

- Client side: Presentation and Data Access layers that run in your browser.
  - The **visual widget library** from the Presentation layer and **JavaScript model library** from the Data access layer have a clear separation to increase their reuse.
- Server side: Service and Backend systems layers.

Components of IBM Content Navigator

Model-View-Controller (MVC) architecture

- View tier - Representation that the user accesses.
- Model tier - Contains the business rules.
- Controller tier - Propagates the information to the model and viewer tier.

Components of IBM Content Navigator

- Visual widget library - Represents the view tier for IBM Content Navigator.
- Modeling library - Defines the model tier for IBM Content Navigator.
- Web client - Acts as a controller.
- Midtier services - Contain specific implementations and interfaces to the repositories.

IBM Content Navigator web client

- Acts as a controller to provide communication between the view and the model.
- Translates the users interactions with the widgets in the client into actions that instances of the model classes do.

IBM Content Navigator visual widget library

- This library of JavaScript classes defines widgets that are used to build the view for IBM Content Navigator. It is built on Dojo 1.8.4
- The following list shows the important IBM Content Navigator widgets:
  - Search form widget: Renders search forms for saved searches.
  - Folder tree widget: Displays the list of folders in a repository.
  - Content viewer widgets: Enable the displaying of documents.
  - Content list widget: Displays documents and folders in a repository and allow users to do the actions on these documents or open them.
  - Teamspace builder widgets: Create and edit a Teamspace in IBM Content Navigator.
  - Workflow widgets: View and edit workflows, work lists, and work items.
- New custom visual widgets can be created in the visual layer.
- Inherit or overwrite an existing widget from the Content Navigator visual widget library.

IBM Content Navigator modeling library

Objective C, JavaScript, .NET

- Decouples the business logic and data from the repository-specific services and the representation tier.
- Specifies communication with the mid-tier services to access the Content repositories.
- Allows you to retrieve, create, and change the content from separate repositories
- Use the modeling JavaScript classes to customize the data exchange between the web client and the repositories.
- Knowing the repositories API is not necessary because this tier offers common request, response, timeout, and session handling.

The following list shows the important JavaScript modeling tier classes:

- **Request class**
  - Represents a single request to the services
- **Repository class**
  - Is an abstraction for the supported repositories of IBM Content Navigator.
  - Contains basic methods for document creation and deletion, content retrieval, and privilege roles setup.
  - Provides repository connection information and the retrieval of search templates.
- **Item class**
  - You can obtain an item from a ResultList.

    Example of an item: folder, document, or a worklist item in the repository.
- **Search and SearchTemplate class**
  - Provides methods to define a new search.
- **Desktop class**
  - One of the important components of the modeling tier.
  - Provides methods to get and set information for the current desktops and other features.

IBM Content Navigator Mid-tier Services

- The **Mid-tier Services** provide the REST services for accessing repositories, the configuration database, or your own custom backend system.
- The interface is not made public to use it directly. But you can do the following tasks:
  - Extend it with your own services.
  - Modify and replace existing services by intercepting the requests and responses messages.
- JavaScript Modeling API uses the mid-tier services.

Develop applications with IBM Content Navigator

IBM Content Navigator provides a powerful platform for building custom web applications to manage content.

- Use extension points and Application Programming Interfaces (APIs) to extend the web client by adding custom actions, menus, layouts, or services.
- Use the APIs to build custom applications that incorporate IBM Content Navigator features without using the standard web client.

Programming requirements

- Java
  - Use the IBM Content Navigator Java API to create plug-ins to implement the functions that you want to add to the web client.
- JavaScript
  - Use the IBM Content Navigator JavaScript API to create custom widgets to use with the plug-ins.
- Web programming
  - To implement the JavaScript functions for the actions that the plug-in defines.
- Dojo Toolkit
  - The toolkit that IBM Content Navigator uses for the visual widgets.

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.

IBM Content Navigator plug-ins

- An IBM Content Navigator plug-in allows you to extend the product to meet specific needs of a particular business.
  - Add functions directly into the Content Navigator user interface.
  - Extend existing functions, or add new features to the application.
- You can create plug-ins to provide the following extensions:
  - A new feature (Search, Browse, and Teamspace are default features)
  - Custom actions and menus (Checkin, Checkout, and Download are default actions)
  - New layouts (A layout defines the arrangement of the widgets that are used in a desktop)
  - Dijits (Example: The Content Navigator tree control, Add document dialog)
  - Custom services (Example: IBM Content Analytics)

  - Intercept and modify the requests or responses sent (Example: External Data Services)
  - Add third-party viewers
- You configure the user interface for setting up the plug-in specific information in the Content Navigator administration tool.

Setting up the development environment for plug-ins

Development environment

You can use any Eclipse-based development environment to extend and customize IBM Content Navigator through plug-ins.

- 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. It includes the basic JEE components, JavaScript, HTML and CSS Editors. Most of the lab exercises in this course use the Eclipse Kepler package that is based on Eclipse Version 4.3.1. For the IBM Worklight Eclipse project for mobile development, you use the Juno Eclipse package.

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 two Java Archive (JAR) files:
  - com.ibm.ecm.plugin.202.jar
  - com.ibm.ecm.icn.facet.EDSPlugin.202.jar
- You can create a new project for IBM Content Navigator plug-in development and a new Eclipse web project facet for building EDS web projects with these JAR files.

Installation of the Eclipse plug-in

1. Download the JAR files from this IBM Redbooks publication’s more materials page.
2. 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

3. Restart Eclipse.

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

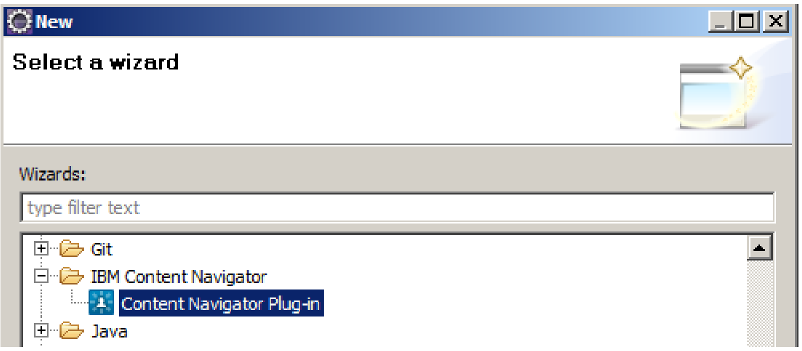
Note

The Eclipse plug-in can be downloaded from the additional material link that is associated with this IBM Redbooks publication. For the student image, these plug-ins are already included in Eclipse.

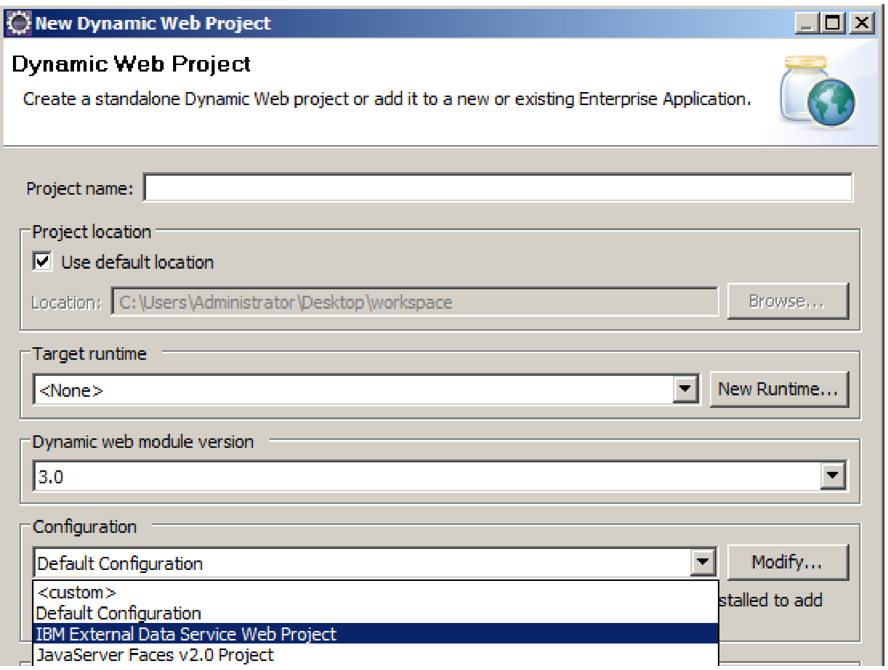
Verification of Eclipse plug-in installation

After the Eclipse plug-in installation, the options to create a “Content Navigator Plug-in” project and a “IBM External Data Service Web Project” are available in Eclipse.

- In Eclipse, select File > New > Project and verify that you are able to see the option for “Content Navigator Plug-in” project.



- In Eclipse, select File > New > Dynamic Web Project. In the New Dynamic Web Project page, verify that you are able to see the option for “IBM External Data Service Web Project” in the list for the Configuration field.



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

☑ Use default location

Location: C:\Users\Administrator\Desktop\workspace

Target runtime: <None>

Dynamic web module version: 3.0

Configuration: Default Configuration

Troubleshooting Eclipse plug-in installation

If you restart Eclipse and do not see the option to create an IBM Content Navigator plug-in project, then you most likely do not have all the dependencies installed.

- Check the Eclipse logs for errors.
- Click the Windows > Show View > Error Log menu.
- A common error in the log looks like the following text:

```
Unable to satisfy dependency from com.ibm.ecm.icn.plugin.202 to bundle
org.eclipse.wst.jsdt.core [1.1.202,2.0.0).
```

- To resolve this error, go to the Eclipse marketplace and download the missing dependency.

Note

If you install Rational Application Developer 8.5 or higher or the latest JEE version of Eclipse, then all IBM Content Navigator Eclipse plug-in dependencies are already installed.

Demonstrations

Create an IBM Content Navigator plug-in project in Eclipse

Click here to watch the demonstration.

Exercise 2.1.1: Create a plug-in project in Eclipse

Introduction

In this exercise, you create a plug-in project in Eclipse with the IBM Content Navigator Eclipse plug-in. You register the plug-in with IBM Content Navigator.

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

Procedure 2: Check the packages and files

Procedure 3: Configure Java Compiler

Procedure 4: Check the generated code in EDUPlugin.java file

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.

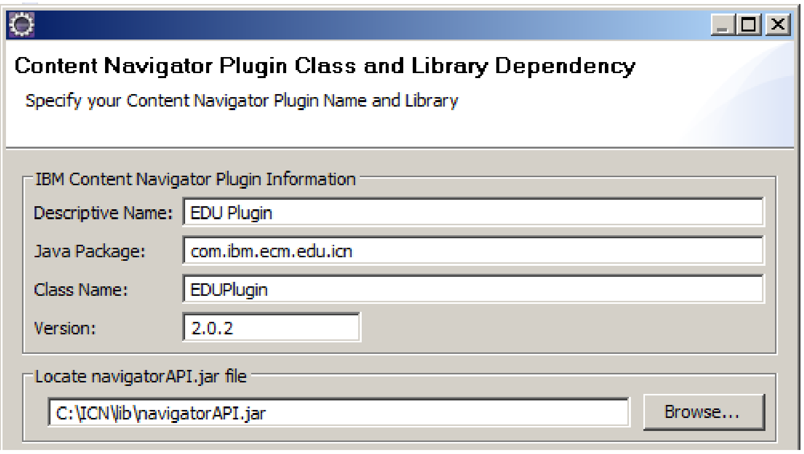
1. Open Eclipse by double-clicking the Eclipse_Kepler icon in your desktop.
   a. In the Workspace Launcher page, leave the default workspace directory (`C:\ICN\workspace_Kepler`) and click OK.
2. Open the project creation wizard.
   a. In Eclipse, click File > New > Other.
   b. In the `New` page, scroll down and select IBM Content Navigator > Content Navigator Plug-in. 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 next 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.ecm.edu.icn	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	2.0.2	Version of the plug-in. The Version number is set in the Class that is provided under Class Name.
Locate navigatorAPI.jar	C:\ICN\lib	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\IBM\ECMClient\lib). In your student system, this file is copied into the C:\ICN\lib folder.



Content Navigator Plugin Class and Library Dependency

Specify your Content Navigator Plugin Name and Library

IBM Content Navigator Plugin Information

Descriptive Name: EDU Plugin

Java Package: com.ibm.ecm.edu.icn

Class Name: EDUPlugin

Version: 2.0.2

Locate navigatorAPI.jar file

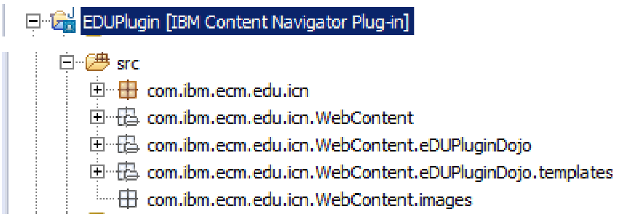
C:\ICN\lib\navigatorAPI.jar

6. Click Finish.
7. When 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.

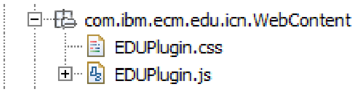


```
EDUPlugin [IBM Content Navigator Plug-in]
  src
    com.ibm.ecm.edu.icn
    com.ibm.ecm.edu.icn.WebContent
    com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo
    com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo.templates
    com.ibm.ecm.edu.icn.WebContent.images
```

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

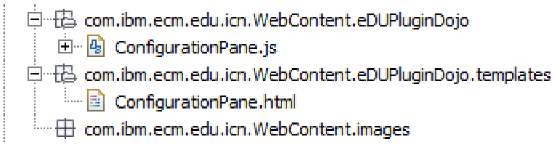
Directories under the src folder of your project	Content
<PackageName> `com.ibm.ecm.edu.icn`	All the Java Classes extending the IBM Content Navigator plug-in classes
<PackageName>/WebContent	CSS, main JavaScript plug-in, and images files
<PackageName>/WebContent/<Dojo>	Dojo classes
<PackageName>/WebContent/<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.ecm.edu.icn.WebContent`)



```
com.ibm.ecm.edu.icn.WebContent
  EDUPlugin.css
  EDUPlugin.js
```

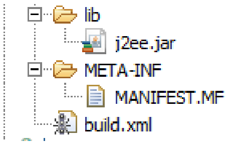
     - Root directory for your plug-in extensions to the client-side of Content Navigator
     - The wizard created shells that you can use to build your customizations.
     - Example: You can add a style change to the CSS file that is generated in this directory.
   - eDUPluginDojo package (`com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo`)



```
com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo
  ConfigurationPane.js
com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo.templates
  ConfigurationPane.html
com.ibm.ecm.edu.icn.WebContent.images
```

     - 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.

  - If you want to prompt the administrator who deploys your plug-in for custom configuration, you must modify the ConfigurationPane JavaScript and HTML template.
- lib folder
  - It contains the j2ee.jar by default.
  - To keep the changes to the build.xml minimum, 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 your project > the MANIFEST file, the parameter “`Plugin-Class`” points to com.ibm.ecm.edu.icn. 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.
  - If you extend the EDUPlugin project to include actions, features, or services, you must add references to the required libraries (Example: repository JAR files) in the path section of the build script.
  - The build.xml also contains the name of the output plug-in JAR file in the JAR section.

Note

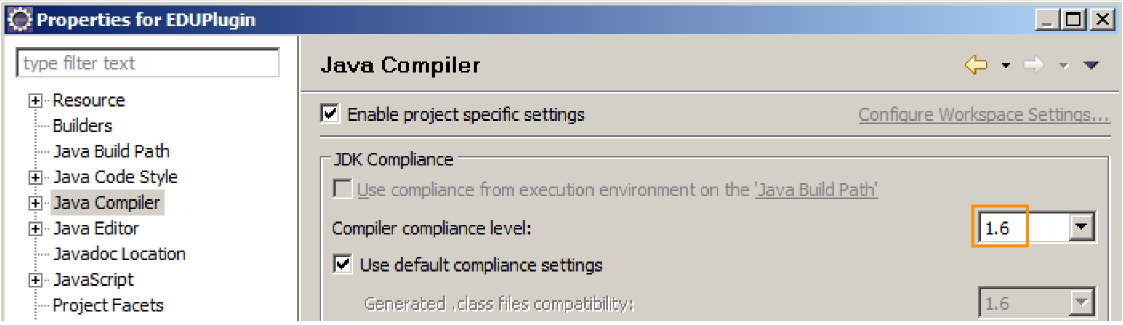
You can also use the Eclipse export function to create a JAR file. This method does not require the build.xml file. But it requires the MANIFEST.MF file.

Procedure 3: 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.



☑ Enable project specific settings

□ Use compliance from execution environment on the 'Java Build Path'

Compiler compliance level: 1.6

☑ Use default compliance settings

Generated .class files compatibility: 1.6

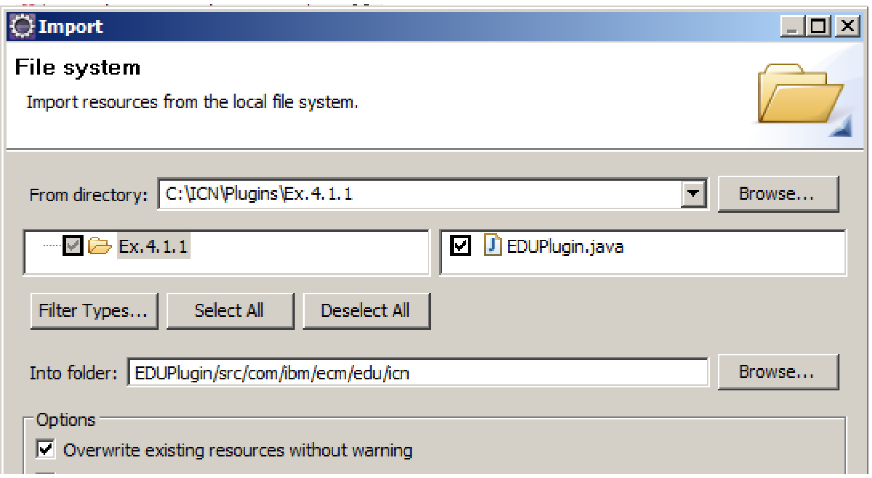
6. When prompted, click Yes in the “Compiler Settings Changed” page to rebuild the project.

Procedure 4: 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. In Package Explorer, expand EDUPlugin > src > com.ibm.ecm.edu.icn.
   a. Double-click the EDUPlugin.java file.
   b. Explore the methods that the wizard generated and their notes in the Java file.
2. Your student image has a copy of the EDUPlugin.java file without notes and extra methods that you can use for the exercises.
   a. Optionally, replace the wizard-generated file with the one in the solution folder with the following steps.
3. Right-click the `com.ibm.ecm.edu.icn` package and click Import from the list.
   a. In the Import page, select General > File System and click Next.
   b. In the Import > File system page, click Browse.
   c. In the “Import from directory” page, go to C:\ICN\Plugins\Ex.4.1.1 folder and click OK.
   d. Back in the Import > File system page, select EDUPlugin.java.

4. Make sure that the “`Into folder`” field has the following value: `EDUPlugin/src/com/ibm/ecm/edu/icn`
   a. Select the “Overwrite existing resources without warning” option.



Import

File system

Import resources from the local file system.

From directory: C:\ICN\Plugins\Ex.4.1.1

☑ Ex.4.1.1

☑ EDUPlugin.java

Into folder: EDUPlugin/src/com/ibm/ecm/edu/icn

Options

☑ Overwrite existing resources without warning

   b. Click Finish.

5. Leave Eclipse open through out this unit.

Note

The project that you created in this exercise is an empty Content Navigator plug-in project. In the following lessons, you add extensions to this project.

Exercise 2.1.2: Register the plug-in with IBM Content Navigator

Introduction

In IBM Content Navigator 2.0.2, the plug-in administration supports the ability to provide a project directory and plug-in class to the Class Loader. This feature allows changes to your plug-in to reflect immediately in IBM Content Navigator, so you can actively test your plug-in code during development. In previous releases, you must package your plug-in as a JAR file before registering it in IBM Content Navigator.

In this exercise, you register your plug-in with IBM Content Navigator with the plug-in class.

Procedures

Procedure 1: Open the IBM Content Navigator Admin Desktop

Procedure 2: Register a plug-in

Procedure 1: Open the IBM Content Navigator Admin Desktop

1. If it is not already started, start the WebSphere Application Server.
   a. Double-click the Start Server.bat file in the WebSphere Admin folder on the desktop.
   b. Wait for the Start the server page to close.
2. In your Firefox browser, go to `http://ecmedu01:9080/navigator/?desktop=admin`.
3. Enter the logon credentials for an administrator.
   - User ID: `p8admin`
   - Password: `IBMFileNetP8`

Procedure 2: Register a plug-in

1. In the Admin desktop, click the Plug-ins icon in the left pane.
2. In the Plug-ins tab, click the New Plug-in button.
   a. In the New Plug-in page, select the `Class file path` option.
   b. Enter the location for the plug-in class file in the File field:

      `C:\ICN\workspace_Kepler\EDUPlugin\bin`

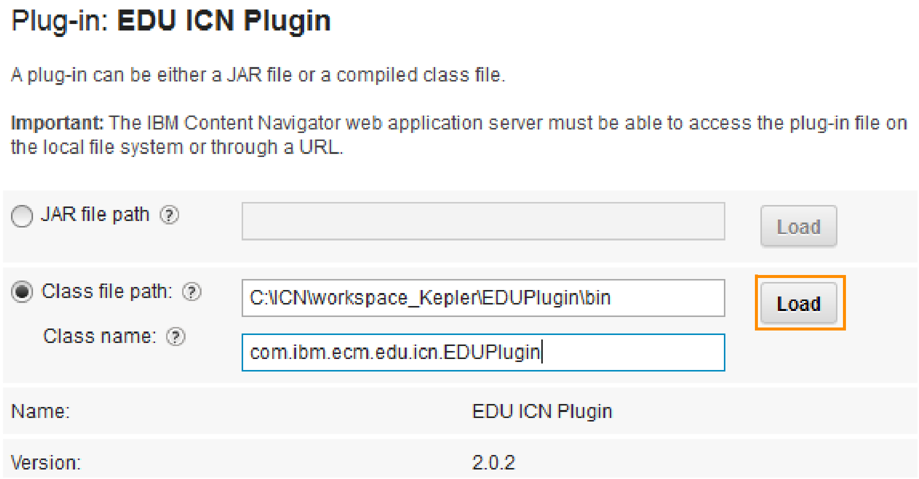
Hint

In Windows Navigator, go to the location of the file. Copy and paste the directory to avoid typing errors.

   c. Enter the class name in the File field: `com.ibm.ecm.edu.icn.EDUPlugin`

d. Click Load.

e. If the file path is valid, the page shows more information as defined for the plug-in.



Plug-in: **EDU ICN Plugin**

A plug-in can be either a JAR file or a compiled class file.

Important: The IBM Content Navigator web application server must be able to access the plug-in file on the local file system or through a URL.

○ JAR file path ? Load

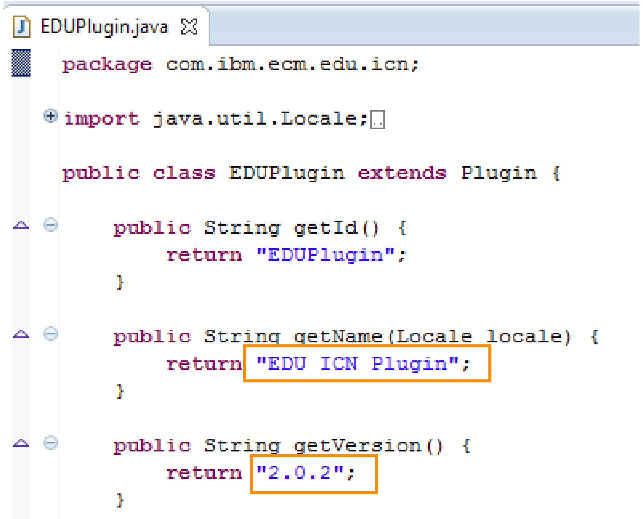
● Class file path: ? C:\ICN\workspace_Kepler\EDUPlugin\bin Load

Class name: ? com.ibm.ecm.edu.icn.EDUPlugin

Name:	EDU ICN Plugin
Version:	2.0.2

3. Observe the lines of code in EDUPlugin.Java file that you just added in Eclipse.

   a. The name and version information in the plug-in Java file is shown in the admin tool.



```
package com.ibm.ecm.edu.icn;

import java.util.Locale;

public class EDUPlugin extends Plugin {

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

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

    public String getVersion() {
        return "2.0.2";
    }
```

   b. Click Save and Close.

   c. Verify that the new plug-in is listed in the Plug-ins tab.

   d. Log out of IBM Content Navigator and close the browser.

Lesson Summary

In this lesson, you completed the following items:

- Learned about the IBM Content Navigator development architecture and components.
- Created a project in Eclipse with Eclipse plug-in for IBM Content Navigator.
- Checked the package and files that the wizard created for your project.
- Registered your plug-in with IBM Content Navigator.

LESSON 2.2: Implement Custom Response Filters

What this lesson is about

In the “Implement External Data Services (EDS)” unit, you used the sample EDS plug-in that IBM Content Navigator provides to manage some of the property behaviors. This lesson shows how to create a plug-in to implement your own custom response filters.

What you should be able to do

After completing this lesson, you should be able to:

- Create your own response filters to add a custom format to a property of an object.

How you will check your progress?

- Hands on Labs.

IBM Content Navigator Plug-ins

Components of a Plug-in

The plug-in consists of the following components that are contained in a single JAR file:

- The web browser logic:
  - Is a required component.
  - Enables users to call the plug-in from the web client.
  - Communicates with the IBM Content Navigator client.
  - Is built with the Content Navigator visual widgets that are based on the Dojo Toolkit.
  - Is implemented in JavaScript.
- The mid-tier server logic:
  - Is a required component of the plug-in.
  - Calls the APIs for the content servers to retrieve the data that the plug-in requires.
  - Communicates with the IBM Content Navigator services on the web application server.
  - Is implemented in Java.
- The configuration component:
  - Is an optional component.
  - Enables an instance of the plug-in to be configured with the IBM Content Navigator administration tool.

Requirements for Plug-ins

- A plug-in must implement a set of abstract Java classes that provide Content Navigator with the following information:
  - The function that the plug-in provides.
  - How the function must integrate with the base product.
- The code for a plug-in is packaged as a single JAR file.
  - The plug-in is registered with Content Navigator through the administration tool.
  - After the registration, the plug-in function is available in the base product.
- A plug-in JAR file is self-contained.
  - The plug-in can be for a single new action or for a set of new services and layout.
- Each plug-in project must contain a Plug-in class that defines the contents of the plug-in.

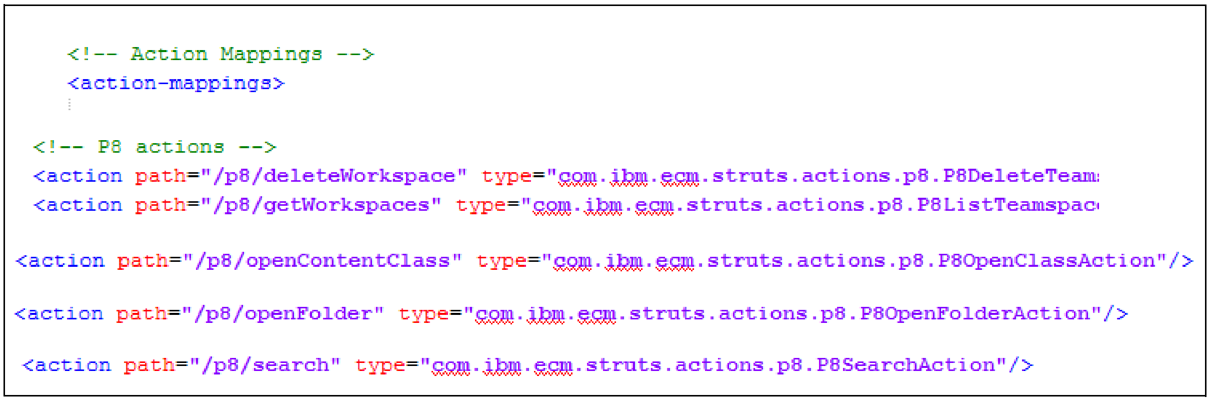
Request and Response filters

Request and response filter overview

A request or response filter stands between the client and the service, and modifies the responses and requests as needed.

- In IBM Content Navigator, when running a service call, a request is sent to the service and the response is sent back from the service to the client tier.
- The IBM Content Navigator framework creates the requests and responses in a standard JSON format.
- The filters enable you to modify:
  - A request before it is sent to the service.
  - A response after it is received from the service.
- A filter is assigned to one service or a list of services within IBM Content Navigator.
  - Provided services are listed in the `struts-config.xml` file, which is in the following directory:

    `<WAS_InstallPath>/profiles/<nameOfProfile>/installedApps/<nameOfCell>/navigator.ear/navigator.war/WEB-INF`
  - The `<action-mappings>` section in the file lists the actions that can be filtered.
  - The `<action path>` tag gives the action name that the `getFilteredServices()` method must return.



```
    <!-- Action Mappings -->
    <action-mappings>
    ⋮

  <!-- P8 actions -->
  <action path="/p8/deleteWorkspace" type="com.ibm.ecm.struts.actions.p8.P8DeleteTeam
  <action path="/p8/getWorkspaces" type="com.ibm.ecm.struts.actions.p8.P8ListTeamspac

<action path="/p8/openContentClass" type="com.ibm.ecm.struts.actions.p8.P8OpenClassAction"/>

<action path="/p8/openFolder" type="com.ibm.ecm.struts.actions.p8.P8OpenFolderAction"/>

 <action path="/p8/search" type="com.ibm.ecm.struts.actions.p8.P8SearchAction"/>
```

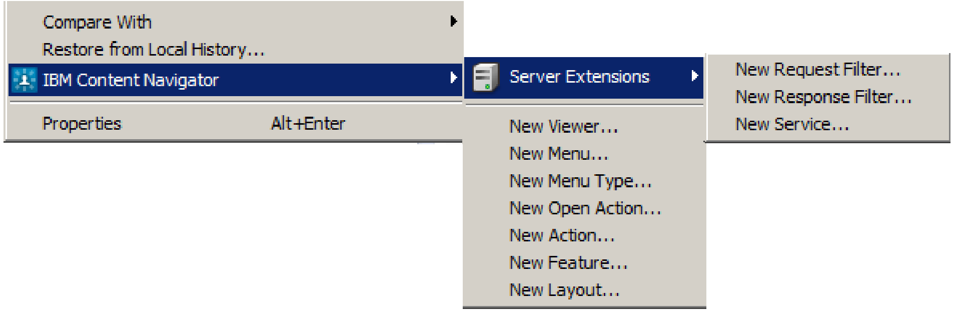
IBM Content Navigator plug-in extensions

Creating server-side extensions

- The Content Navigator plug-in menu in Eclipse provides wizards to add more plug-in components such as Viewer, Menu, Action, Feature, and Layouts.
  - This tool also enables you to create several server-side extensions such as Request Filters, Response Filters, and Services to your plug-in.
  - Server Extensions apply only to the IBM Content Navigator server-side, while other plug-in extensions can have client-side enhancements as well.

Creating response filters

- Response filters are extensions to existing IBM Content Navigator services, which allow you to manipulate the standard JavaScript Object Notation (JSON) payload that the server-side service returns.
- You can use this feature to insert custom formatter widgets in the list and property information user interface widgets, alter values within the JSON or insert entirely new entries in the JSON return.



Creating request filters

In addition to response filters, you can also create request filters and services.

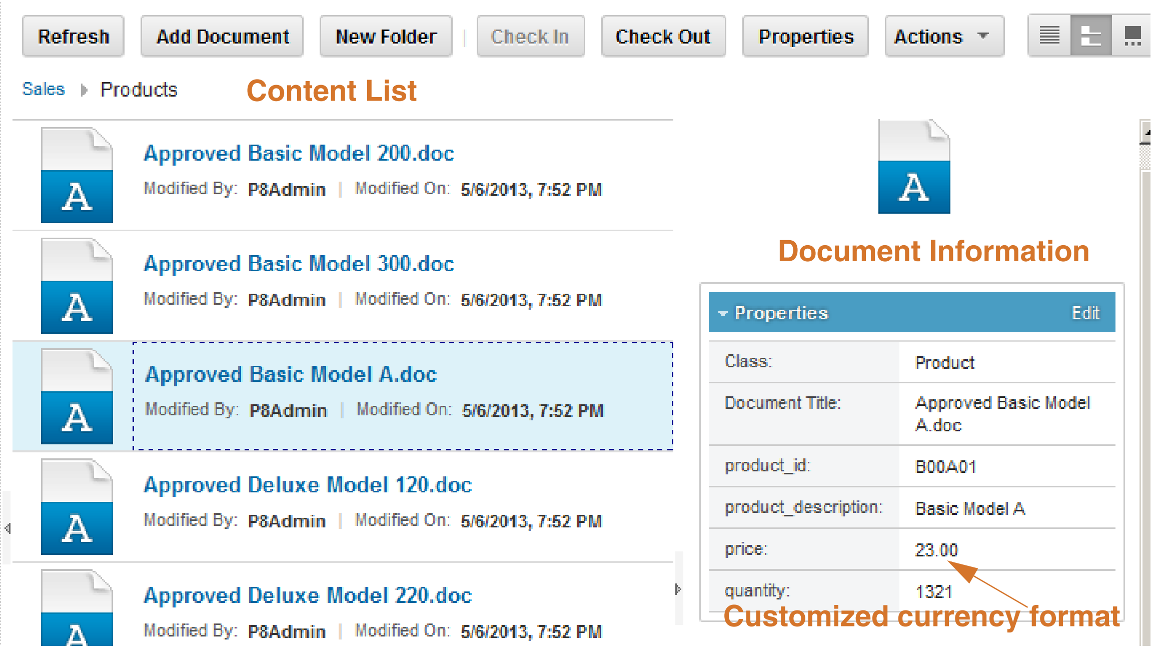
- Request filters are similar to response filters, but instead of applying after the IBM Content Navigator services completed, a request filter is applied before the service runs.
  - A request filter allows the plug-in to manipulate the request parameters.
  - You create a plug-in service if you need a new server-side procedure to IBM Content Navigator.
  - Example: you can use this method to make more capabilities available from an Enterprise Content Manager repository that are not yet available by default.

Content List and Document Information areas for this lesson

In this lesson exercises, you implement custom response filter that changes the format of the properties that are shown in the Content List and Document information areas.

Content List and Document Information area in IBM Content Navigator

The screen capture shows the Content List and Document information area in IBM Content Navigator. The customized currency formatting for the `price` property (integer value) is shown.



Content List

Approved Basic Model 200.doc
Modified By: P8Admin | Modified On: 5/6/2013, 7:52 PM

Approved Basic Model 300.doc
Modified By: P8Admin | Modified On: 5/6/2013, 7:52 PM

Approved Basic Model A.doc
Modified By: P8Admin | Modified On: 5/6/2013, 7:52 PM

Approved Deluxe Model 120.doc
Modified By: P8Admin | Modified On: 5/6/2013, 7:52 PM

Approved Deluxe Model 220.doc
Modified By: P8Admin | Modified On: 5/6/2013, 7:52 PM

Document Information

Properties Edit

Class:	Product
Document Title:	Approved Basic Model A.doc
product_id:	B00A01
product_description:	Basic Model A
price:	23.00
quantity:	1321

Customized currency format

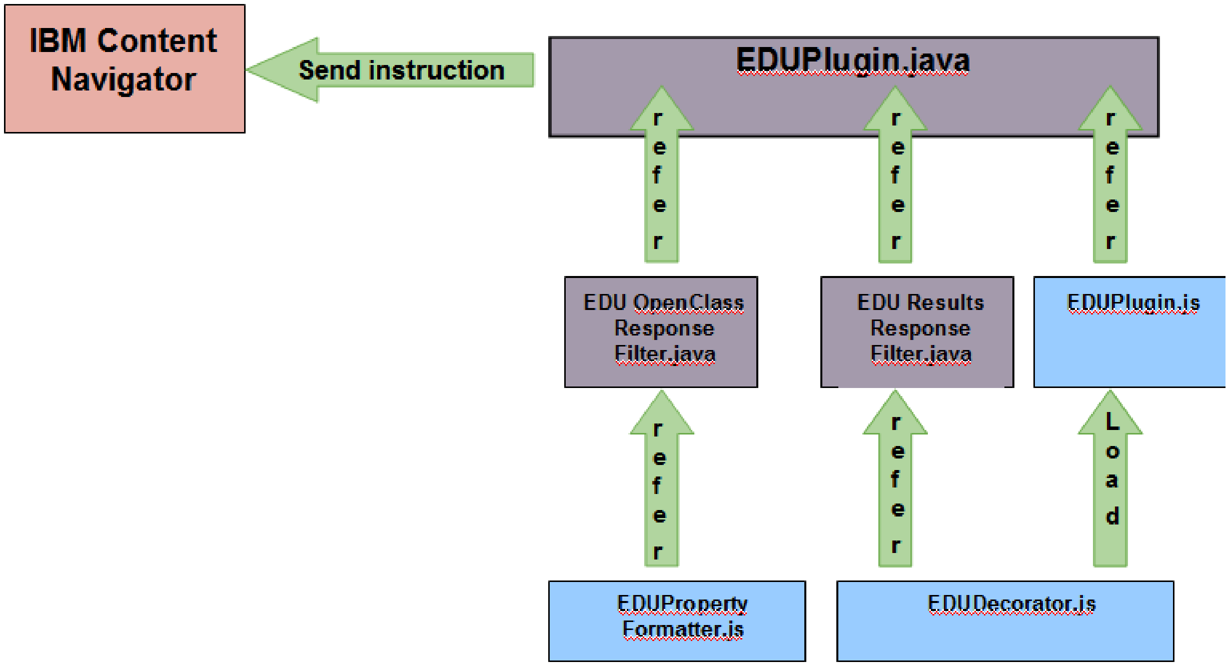
ContentList widget

- A Dojo widget in the IBM Content Navigator toolkit.
  - Class: `ecm.widget.listView.ContentList`
- Responsible for displaying lists of objects in the IBM Content Navigator web application.
  - Examples: search results, the contents of a folder, work items, lists of teamspace.

Document information view

- A module in the ContentList.
- Used to display property information when you select an object in the ContentList.
- The Document information Dojo widget provides a mechanism for customizing property display, similar to the ContentList.

Files that are used for this lesson



Java Files

- The following Java files enable custom formatters in the ContentList and Document information area:
  - EDUPlugin.java
  - EDUOpenClassResponseFilter.java
  - EDUResultsResponseFilter.java
- These three Java files enable the response filters for the search, open folder, and open class actions in IBM Content Navigator.
- A PluginResponseFilter is an extension that allows customization of the JSON response from the IBM Content Navigator server.
  - You modify the structure of the ContentList to apply a "decorator" to the "price" property display.
  - A decorator is a JavaScript method that alters the display of a property within the ContentList.
  - The response filter also applies a custom "propertyFormatter" to the property display in the document information area.
  - The "propertyFormatter" is JavaScript method that formats the display of a value in the document information property grid.

JavaScript Files

- EDUPlugin.js
  - The base JavaScript file that 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.
  - In this lab, you use this file to load the decorator JavaScript, which provides global methods that are used within the ContentList to format your custom property.
- EDUDecorator.js
  - Custom formats the value of the "price" property that is shown in the ContentList
- EDUPropertyFormatter.js
  - Custom formats the value of the "price" property that is shown in the Document information area

Demonstrations

Implement Custom Response Filters

Click here to watch the demonstration.

Exercise 2.2.1: Implement custom Response Filters

Introduction

The plug-in that you are going to create extends the Content List and Document Information areas of Content Navigator with custom formatting. In this exercise, you add a customization that formats numbers in a property as currency.

Procedures

Procedure 1: Create a Response Filter

Procedure 2: Edit the EDUOpenClassResponseFilter.java file

Procedure 3: Add the EDUResultsResponseFilter.java file

Procedure 5: Import files into your project

Procedure 6: Validate your plug-in

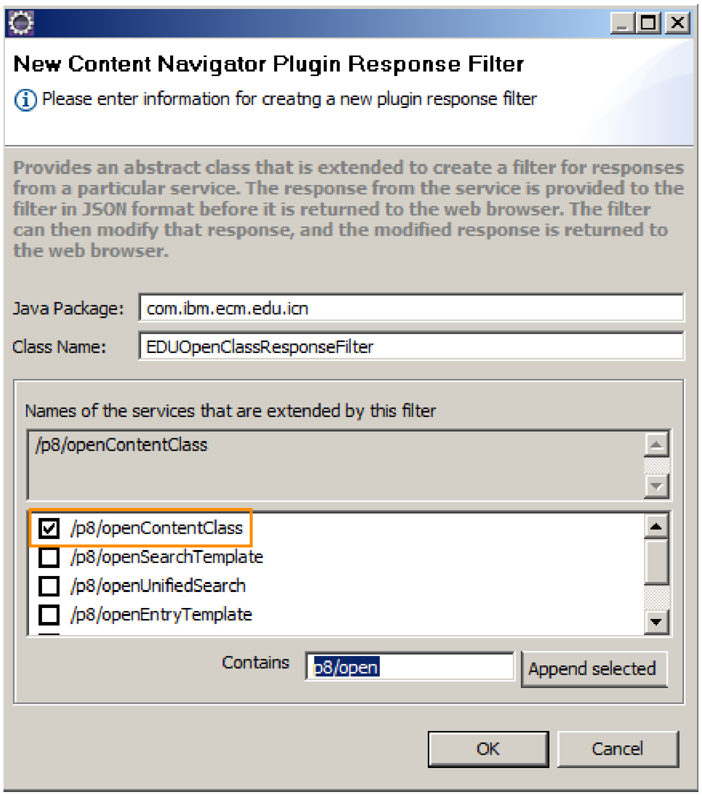
Procedure 7: Troubleshooting

Procedure 1: Create a Response Filter

1. Right-click the package (`com.ibm.ecm.edu.icn`) in your plug-in project and select the IBM Content Navigator > Server Extensions > New Response Filter menu item.
2. In the New Content Navigator Plugin Response Filter page, the wizard fills your Java Package name automatically.
   a. Enter `EDUOpenClassResponseFilter` for the Class Name field.
3. Add the services that are responsible for handling “open content” actions in the IBM FileNet P8 repository to your response filter.
   a. To search for this item, enter p8/open in the `Contains` field to find the items that starts with p8/open in the list.
   b. Select `/p8/openCotentClass` and click “Append selected”.
   c. Click OK.

Note

This interface allows you to select multiple items that match a criteria and add them to your Java class.



New Content Navigator Plugin Response Filter

ⓘ Please enter information for creatng a new plugin response filter

Provides an abstract class that is extended to create a filter for responses from a particular service. The response from the service is provided to the filter in JSON format before it is returned to the web browser. The filter can then modify that response, and the modified response is returned to the web browser.

Java Package: com.ibm.ecm.edu.icn

Class Name: EDUOpenClassResponseFilter

Names of the services that are extended by this filter

/p8/openContentClass

- ☑ /p8/openContentClass
- □ /p8/openSearchTemplate
- □ /p8/openUnifiedSearch
- □ /p8/openEntryTemplate

Contains p8/open

The wizard generates the new response filter Java class (EDUOpenClassResponseFilter.java) and updates the primary plug-in Java class.

Note

By adding the openContentClass service to the list of services that this response filter extends, you instruct IBM Content Navigator to call this response filter for any openCotentClass service request against an IBM FileNet Content Manager (P8). You can also choose services that can instruct multiple repository sources such as P8, IBM Content Manager, or Content Management Interoperability Services.

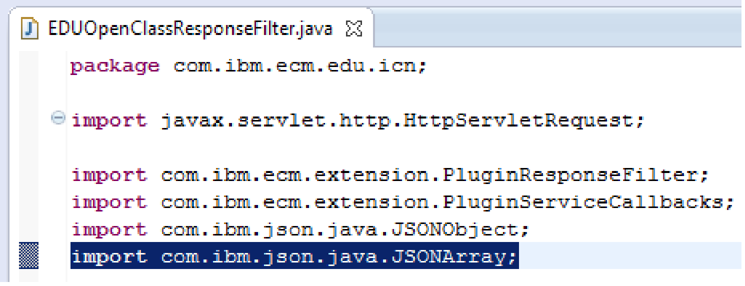
Procedure 2: Edit the EDUOpenClassResponseFilter.java file

You are going to create a response filter for the Product class > price property. This Java file enables the response filter for the openContentClass action in IBM Content Navigator.

1. In Package Explorer, expand EDUPlugin > src > com.ibm.ecm.edu.icn.
   a. Double-click the EDUOpenClassResponseFilter.java file.
2. Observe the packages that the wizard added in the `import` section.

3. Expand the import by clicking the plus sign and add the following package:

   `import com.ibm.json.java.JSONArray;`



```
package com.ibm.ecm.edu.icn;

import javax.servlet.http.HttpServletRequest;

import com.ibm.ecm.extension.PluginResponseFilter;
import com.ibm.ecm.extension.PluginServiceCallbacks;
import com.ibm.json.java.JSONObject;
import com.ibm.json.java.JSONArray;
```

4. The plug-in wizard generates two methods and notes for each method.

   `getFilteredServices()` and `filter(...)`

5. Add the following code (in bold) to the `filter(...)` method.

   This method applies the filter to the Product class > price property and uses the eduPropertyFormatter Dojo class that you add in the following procedure.

Note

The solution file (`EDUOpenClassResponseFilter.java`) is available in the C:\ICN\Plugins\Ex.4.2.1 folder. Optionally, you can copy and paste the text to avoid typing errors.

```
public void filter(String serverType, PluginServiceCallbacks callbacks,
HttpServletRequest request, JSONObject jsonResponse) throws Exception {
    String templateName = request.getParameter("template_name");
    if (templateName.equalsIgnoreCase("Product")) {
        JSONArray properties = (JSONArray) jsonResponse.get("criterias");

        for (int i = 0; i < properties.size(); i++) {
            JSONObject jsonPropDef = (JSONObject) properties.get(i);
            String name = (String) jsonPropDef.get("name");

            if (name != null && name.equals("price")) {
                jsonPropDef.put("propertyFormatter",
                "eDUPluginDojo/eduPropertyFormatter");
                break;
            }
        }
    }
}
```

6. Save and close the file.

Procedure 3: Add the EDUResultsResponseFilter.java file

This response filter file is similar to the one that you edited in the previous procedure. This Java file also creates a response filter for the Product class > price property. This file enables the response filters for the search and open folder actions in IBM Content Navigator.

1. Right-click the `com.ibm.ecm.edu.icn` package and click Import from the list.
   a. In the Import page, select General > File System and click Next.
   b. In the Import > File system page, click Browse.
   c. In the “Import from directory” page, go to C:\ICN\Plugins\Ex.4.2.1 folder and click OK.
   d. Back in the Import > File system page, select `EDUResultsResponseFilter.java`.
2. Make sure that the “`Into folder`” field has the following value: `EDUPlugin/src/com/ibm/ecm/edu/icn`
   a. Select the “Overwrite existing resources without warning” option.
   b. Click Finish.

Procedure 4: Edit the EDUPlugin.java file

You must refer the DojoModule and EDUResultsResponseFilter.java and the EDUPlugin.js files in the main plug-in Java file. The EDUPlugin.java instructs IBM Content Navigator about the available filters.

1. In Package Explorer, expand EDUPlugin > src > com.ibm.ecm.edu.icn.
   a. Double-click the `EDUPlugin.java` file.
   b. Add code to the `getResponseFilters()` method at the end of the file.
   c. Edit the code as shown in the following text. Bolded text indicates the edits.
   d. Make sure that you add a comma and then enter the bolded line of code.

Note

When you added `EDUOpenClassResponseFilter` with the wizard, it updated that value automatically in the `EDUPlugin.java` file. Since you added the `EDUResultsResponseFilter` without the wizard, this value needs to be added manually in the `EDUPlugin.java` file.

```
public com.ibm.ecm.extension.PluginResponseFilter[] getResponseFilters() {
  return new com.ibm.ecm.extension.PluginResponseFilter[] {
    new com.ibm.ecm.edu.icn.EDUOpenClassResponseFilter(),
    new com.ibm.ecm.edu.icn.EDUResultsResponseFilter()
    };
}
```

Adding the package name (`com.ibm.ecm.edu.icn`)in the code is optional because the files are in the same package. You can add: new `EDUResultsResponseFilter()`

2. Add the code as shown in the following text. Bolded text indicates the required edits.

   The letter “e” in “eDUPluginDojo” must be in lowercase.

> Note
>
> When you created the project with the wizard, it added this part of code automatically to the EDUPlugin.java file. The EDUPlugin.java file that you replaced in the previous lesson does not have these lines of code. You must add the lines manually.

```
public String getVersion() {
  return "2.0.2";
}
public String getScript() {
  return "EDUPlugin.js";
}
public String getDojoModule() {
  return "eDUPluginDojo";
}
public PluginResponseFilter[] getResponseFilters() {
```

> Note
>
> The solution file (EDUPlugin.java) is available in the C:\ICN\Plugins\Ex.4.2.1 folder. Optionally, you can copy and paste the text to avoid typing errors.

3. Save and close the file.

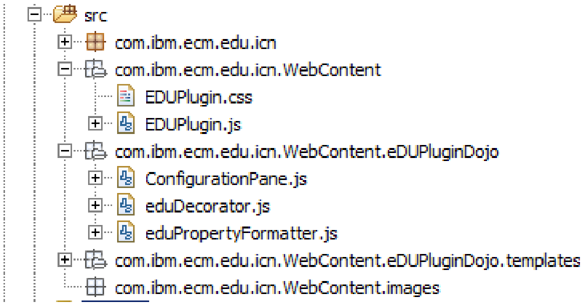
Procedure 5: Import files into your project

1. Import the files from the C:\ICN\Plugins\Ex.4.2.1 folder into the packages with the data in the following table.
2. Make sure that the “Into folder” field has the correct package.
   a. Select the “Overwrite existing resources without warning” option.

Package name in your project	**Files to import**
com.ibm.ecm.edu.icn.WebContent	EDUPlugin.js
com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo	eduDecorator.js eduPropertyFormatter.js

   b. Click Finish.

3. Verify that the files are listed in the Package Explorer.
4. Your completed project must look like the following screen capture:

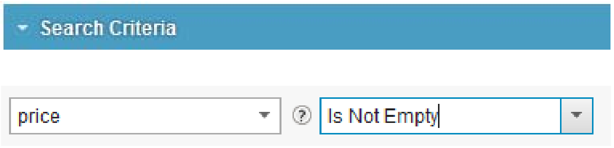


```
src
  com.ibm.ecm.edu.icn
  com.ibm.ecm.edu.icn.WebContent
    EDUPlugin.css
    EDUPlugin.js
  com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo
    ConfigurationPane.js
    eduDecorator.js
    eduPropertyFormatter.js
  com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo.templates
  com.ibm.ecm.edu.icn.WebContent.images
```

5. Open each file and observe the code.
6. The `EDUPlugin.js` file is the base JavaScript file that is loaded when IBM Content Navigator loads the plug-in.
   – In this lab, it loads the decorator JavaScript, which provides global methods that are used within the ContentList to format your custom property.
7. The `eduPropertyFormatter.js` and `eduDecorator.js` files applies the currency format to the “price” property in IBM Content Navigator Details View.

Procedure 6: Validate your plug-in

1. Check how the values of the price property (Product class) are shown in the Sample Desktop before implementing your custom response filter.
2. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
3. Create a search.
   a. Select the “Search” view by clicking the magnifying glass icon in the navigation bar on the left side of the user interface.
   b. Verify that Sales repository is selected.
   c. Click “New Search” at the top of the search selector area.
   d. On the New Search tab, open the class selector and select the `Product` class.
   e. Select the `price` property and select “Is Not Empty” as the search option.

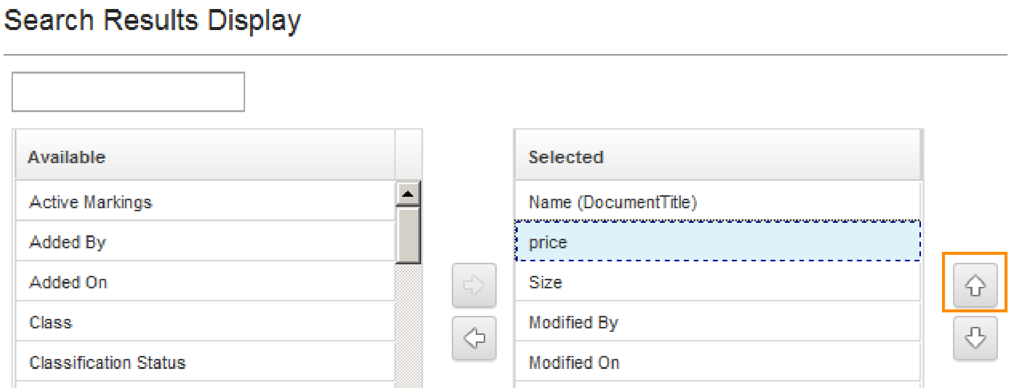


price | Is Not Empty

   f. Click “Results Display”.

g. In the Search Results Display, move the `price` property to the selected pane.

h. Select `price`. Use the up arrow on the right to move the price up next to the Name.



Search Results Display

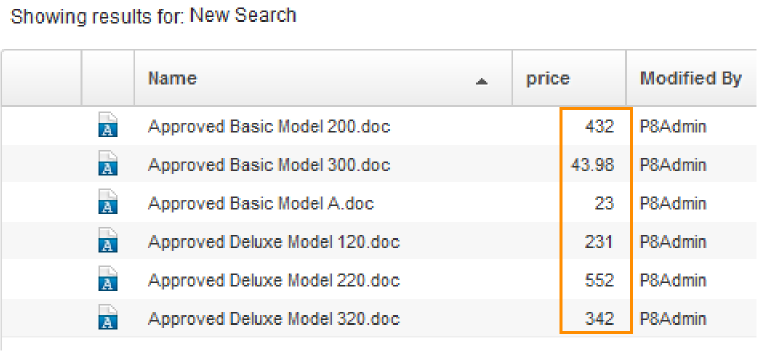
Available
- Active Markings
- Added By
- Added On
- Class
- Classification Status

Selected
- Name (DocumentTitle)
- price
- Size
- Modified By
- Modified On

i. Click OK.

4. Click “Search”.

5. Check how the values for price property are shown before implementing the response filter.



Showing results for: New Search

	Name	price	Modified By
A	Approved Basic Model 200.doc	432	P8Admin
A	Approved Basic Model 300.doc	43.98	P8Admin
A	Approved Basic Model A.doc	23	P8Admin
A	Approved Deluxe Model 120.doc	231	P8Admin
A	Approved Deluxe Model 220.doc	552	P8Admin
A	Approved Deluxe Model 320.doc	342	P8Admin

6. Select the “Administration” view by clicking the “gear” icon in the navigation bar on the left side of the user interface.

7. In the Admin desktop, click the Plug-ins icon in the left pane.

8. In the Plug-ins tab, select the `EDU ICN Plugin` and click Edit.

   a. Make sure that the values are already filled for the class file path and class name.

   b. Click Load to reload the plug-in with the changes.

   c. Click Save and Close.

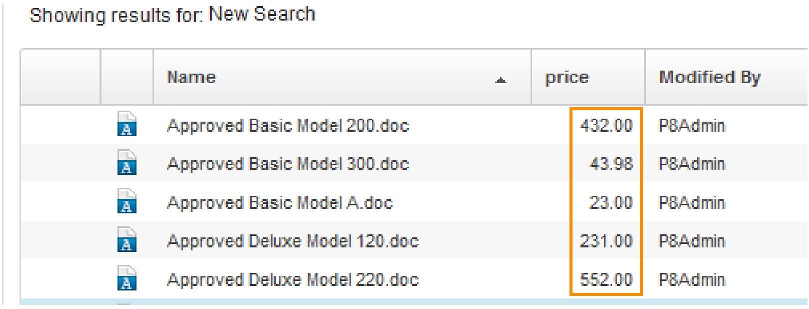
9. Log out of Content Navigator and close the browser.

10. Check the format for the `price` property values after implementing your response filter.

11. Repeat Steps 2 - 5 to search for the documents with the `price` property (Product class).

12. In the search results, verify that the `price` property values are in the currency format after you implemented the custom response filter.

    The response filter that is assigned to the `search` action in your plug-in does the format change for this view.

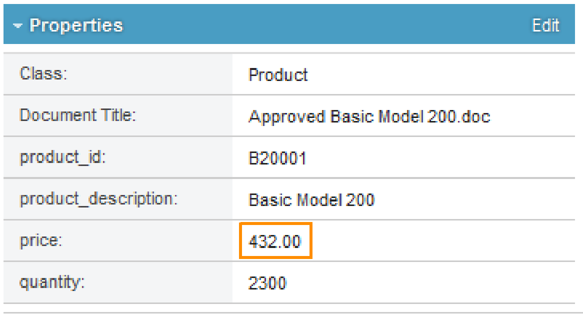


Showing results for: New Search

		Name	price	Modified By
	A	Approved Basic Model 200.doc	432.00	P8Admin
	A	Approved Basic Model 300.doc	43.98	P8Admin
	A	Approved Basic Model A.doc	23.00	P8Admin
	A	Approved Deluxe Model 120.doc	231.00	P8Admin
	A	Approved Deluxe Model 220.doc	552.00	P8Admin

13. Select a document to see the custom format in the document information properties section.

    The response filter that is assigned to the open`ContentClass` action in your plug-in does the format change for this view

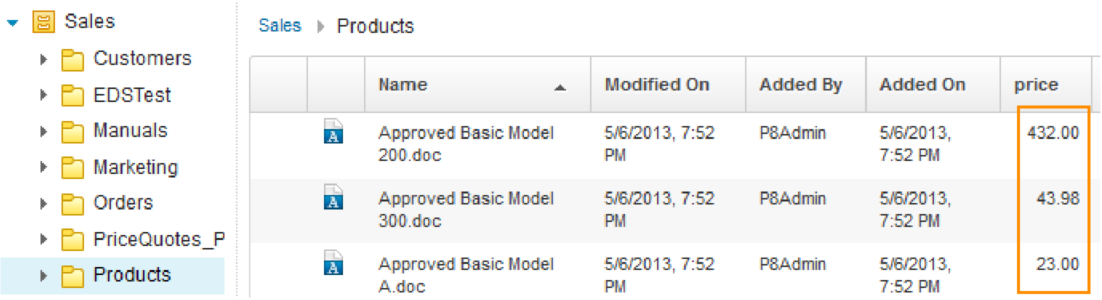


Class:	Product
Document Title:	Approved Basic Model 200.doc
product_id:	B20001
product_description:	Basic Model 200
price:	432.00
quantity:	2300

14. Go to the Browse view > Sales > Products where the `Product` class documents are listed.

    a. Verify that the values for price property are in the custom format.

    The response filter that is assigned to the open`Folder` action in your plug-in does the format change for this view.



		Name	Modified On	Added By	Added On	price
	A	Approved Basic Model 200.doc	5/6/2013, 7:52 PM	P8Admin	5/6/2013, 7:52 PM	432.00
	A	Approved Basic Model 300.doc	5/6/2013, 7:52 PM	P8Admin	5/6/2013, 7:52 PM	43.98
	A	Approved Basic Model A.doc	5/6/2013, 7:52 PM	P8Admin	5/6/2013, 7:52 PM	23.00

15. Log out of the Content Navigator and close the browser.

Procedure 7: Troubleshooting

Troubleshooting

If the update that you made is not reflected in the Content Navigator web client, do the following steps.

The wizard created the ConfigurationPane.js file. Sometimes, the Dojo module reference must be updated.

1. Edit the ConfigurationPane.js file.
   a. In Eclipse > Package Explorer, expand EDUPlugin > src > com.ibm.ecm.edu. icn.WebContent. eDUPluginDojo
   b. Double-click the `ConfigurationPane.js` file.
   c. Edit the code as shown in the following text. Bolded text indicates the edits.

      ```
      function(declare, _TemplatedMixin, _WidgetsInTemplateMixin,
      PluginConfigurationPane, template) {
      ```

      ```
      return declare("eDUPluginDojo/ConfigurationPane", [
      PluginConfigurationPane, _TemplatedMixin, _WidgetsInTemplateMixin], {
      ```
   d. Save and close the file.
   e. Reload the plug-in and test.
2. If Step 1 does not fix the issue, refer to the solution and verify the code in your files.
   a. Replace with the solution files if needed.
3. In the Administration desktop, delete the plug-in and create a new one to make sure the plug-in reloads.
4. Use Firebug to debug (refer to the appendix for more details).

Lesson Summary

In this lesson, you completed the following tasks:

- Learned about IBM Content Navigator plug-ins and Request and Response filters.
- Created server-side extensions.
  - Created a Response Filter.
  - Created files that are needed for your plug-in.
  - Validated your plug-in.

LESSON 2.3: Create a Custom Property Editor

What this lesson is about

In this lesson, you make changes to the custom plug-in that you developed in the previous lessons. You extend an IBM Content Navigator dijit to create a custom property editor for an object class property.

With custom property editors, you can control user input while adding and editing documents.

What you should be able to do

After completing this lesson, you should be able to:

- Create a custom property editor.

How you will check your progress?

- Hands on labs.

IBM Content Navigator terms that are used in this lesson

Widgets

- A widget is a user interface component to provide a specific function that can be used in constructing a user interface.
- Widgets are typically packaged in a library of related components.
- A designer can construct a web page from multiple widgets that work together to provide a unified visual experience to users.
- A widget can be a visual widget that is responsible for a portion of the interface that users see such as a tree view.
- It can be a non-visual widget, which provides supporting function to the user interface, such as form validation.

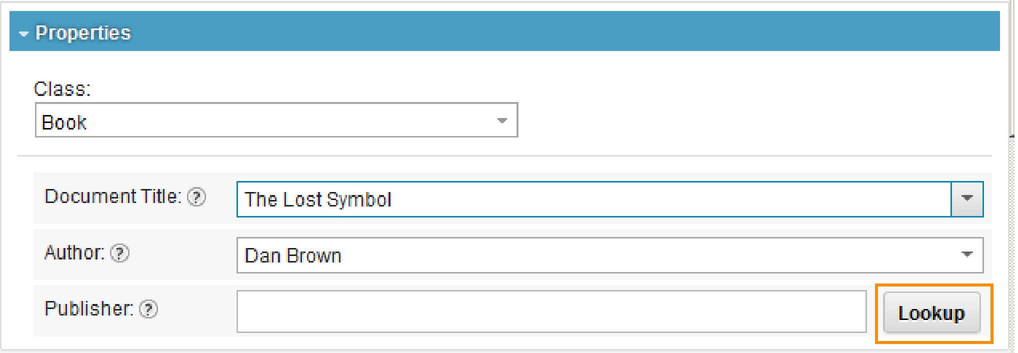
Dojo, Dojo widgets (dijits)

- A Dojo is a JavaScript library of widgets that is designed to increase speed and ease of development of client-side code for a web application.
- Dojo widgets are both configurable and extensible.
- The word dijit is an abbreviation for a Dojo widget.
- Dijit layout: A dijit layout is a dijit that contains child widgets. It is a JavaScript class that extends dijit/_Container class or its subclasses.

Custom Property Editors

With custom property editors, you can control user input while adding and editing documents.

- In this lesson, you create a custom property editor (Dojo widget) for “Publisher” property of the Book class for your custom plug-in.
  - This feature provides a mechanism to override the default IBM Content Navigator property editors.
  - It provides a button next to the standard input text area for a user lookup.
  - When a user clicks this button, it brings up a user lookup dialog. The user must use this new dialog for finding and adding a user name to the “Publisher” attribute.



Properties

Class: Book

Document Title: The Lost Symbol

Author: Dan Brown

Publisher:

Files that are used for this lesson

Java Files

- The following Java files enable the custom property editor in the Add Document wizard or the Property Edit tool:
  - EDUPlugin.java
  - EDUOpenClassResponseFilter.java
- A PluginResponseFilter is an extension that allows customization of the JSON response from the IBM Content Navigator server.

JavaScript Files

- EDUPropertyEditor.js
  - Provides the custom property editor for the "Publisher" property.
  - This file is a Dojo widget that is an extension of the IBM Content Navigator ValidationTextBox Dojo widget.

Other Files

- EDUPropertyEditor.html
  - A template is an HTML file that provides the base layout for a Dojo widget.
  - IBM Content Navigator uses the dijit._TemplatedMixin to build widgets that helps you to create widgets quickly and easily.
  - This template adds a button next to the input box in the browser.
- EDUPlugin.css
  - This file adds a set of styles that are required to position the button next to the text input in the browser.

Exercise 2.3.1: Add a custom property editor

Introduction

In this lab exercise, you create a custom property editor for the “Publisher” property of the Book class and add it to the custom plug-in that you developed in the previous lessons.

You extend an IBM Content Navigator ValidationTextBox dijit (Dojo widget) to create a custom property editor.

Procedures

Procedure 1: Import the Property Editor JavaScript class

Procedure 2: Import the Property Editor template

Procedure 3: Edit or import the Style Sheet

Procedure 4: Edit the EDUPlugin.java file

Procedure 5: Edit the EDUOpenClassResponseFilter.java file

Procedure 6: Validate your plug-in

Procedure 1: Import the Property Editor JavaScript class

In this procedure, you add the eduPropertyEditor.js file to the custom plug-in.

1. In Package Explorer, expand EDUPlugin > src.
2. Right-click `com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo` and click Import from the list.
3. 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:\ICN\Plugins\Ex.4.3.1 folder and click OK.
4. Back in the Import > File system page, select `eduPropertyEditor.js`.
   a. Make sure that the “`Into folder`” field has the following value: `EDUPlugin/src/com/ibm/ecm/edu/icn/WebContent/eDUPluginDojo`
   b. Select the “Overwrite existing resources without warning” option.
5. Click Finish.
6. Verify that the file is added in the eDUPluginDojo package and open the file.
7. In the “`define`” section at the top of the file, check that the `ecm/widget/ValidationTextBox` and `ecm/widget/dialog/SelectUserGroupDialog` Content Navigator Dojo widgets are defined.
8. Notice that the EDUPropertyEditor.html template is also defined. You add this file in the following procedure.

Procedure 2: Import the Property Editor template

1. Right-click `com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo.templates` and 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:\ICN\Plugins\Ex.4.3.1 folder and click OK.
3. Back in the Import > File system page, select `EDUPropertyEditor.html`.
   a. Make sure that the "`Into folder`" field has the following value: `EDUPlugin/src/com/ibm/ecm/edu/icn/WebContent/eDUPluginDojo/templates`
   b. Select the "Overwrite existing resources without warning" option.
4. Click Finish.
5. Verify that the file is added in the templates package.

Procedure 3: Edit or import the Style Sheet

Note

When you created the project, the wizard added an empty `EDUPlugin.css` file to your project. For this procedure, you can copy and paste the code to this file from the solution. To make it easy, you can import the solution file to replace the wizard-generated file.

1. Right-click `com.ibm.ecm.edu.icn.WebContent` and 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:\ICN\Plugins\Ex.4.3.1 folder and click OK.
3. Back in the Import > File system page, select `EDUPlugin.css`.
   a. Verify that the "`Into folder`" field has the following value: `EDUPlugin/src/com/ibm/ecm/edu/icn/WebContent`
   b. Make sure that you select the "Overwrite existing resources without warning" option.
4. Click Finish.
5. Verify that the file is added in the WebContent package.
6. Open the file and check that it contains the styles.

Procedure 4: Edit the EDUPlugin.java file

You must refer the Style Sheet, Dojo module, EDUOpenClassResponseFilter, and the EDUPlugin.js files in the main plug-in Java file. The EDUPlugin.java instructs IBM Content Navigator about their availability.

Note

When you added EDUOpenClassResponseFilter to your project with the wizard in the previous lesson, the wizard updated that value automatically in the EDUPlugin.java file. You already added the EDUPlugin.js and Dojo module values also to the EDUPlugin.java file in the previous lesson.

1. In Package Explorer, expand EDUPlugin > src > com.ibm.ecm.edu.icn.
   a. Double-click the `EDUPlugin.java` file.
2. Add the code as shown in the following text. Bolded text indicates the required edits.

```
public String getDojoModule() {
    return "eDUPluginDojo";
}
@Override
public String getCSSFileName() {
    return "EDUPlugin.css";
}
public PluginResponseFilter[] getResponseFilters() {
```

Note

The solution file (`EDUPlugin.java`) is available in the C:\ICN\Plugins\Ex.4.3.1 folder. Optionally, you can copy and paste the text to avoid typing errors or you can import the solution file to replace the existing one.

3. Save and close the file.

Procedure 5: Edit the EDUOpenClassResponseFilter.java file

The edit that you make in this procedure updates the OpenClass JSON response. It ensures that the new custom property editor is enabled for the Book class > Publisher property.

1. In Package Explorer, expand EDUPlugin > src > com.ibm.ecm.edu.icn.
   a. Double-click the `EDUOpenClassResponseFilter.java` file.
2. Add the following code (in bold) to the `filter(...)` method.
   a. Remove this line from the code: **`break;`**

Note

The solution file (`EDUOpenClassResponseFilter.java`) is available in the C:\ICN\Plugins\Ex.4.3.1 folder. Optionally, you can copy and paste the text to avoid typing errors or you can import the solution file to replace the existing one.

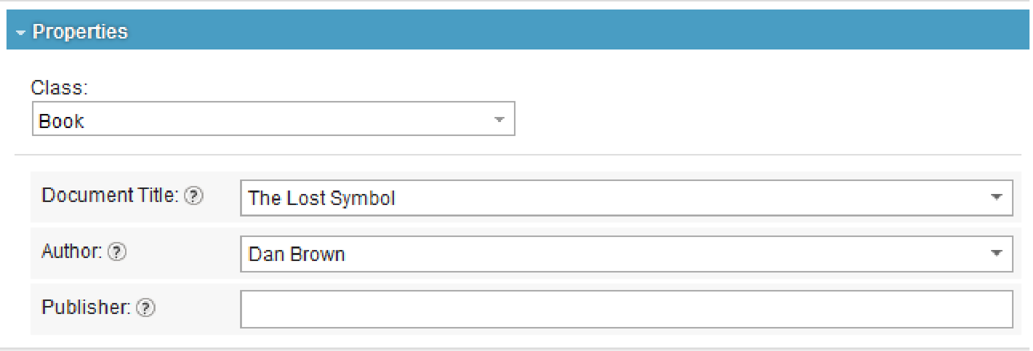
```
      if (name != null && name.equals("price")) {
        jsonPropDef.put("propertyFormatter",
        "eDUPluginDojo/eduPropertyFormatter");
      }
    }
  }
  if (templateName.equalsIgnoreCase("Book")) {
    JSONArray properties = (JSONArray) jsonResponse.get("criterias");

    for (int i = 0; i < properties.size(); i++) {
      JSONObject jsonPropDef = (JSONObject) properties.get(i);
      String name = (String) jsonPropDef.get("name");

      if (name != null && name.equals("Publisher")) {
        jsonPropDef.put("dataType", "xs:user");
        jsonPropDef.put("propertyEditor", "eDUPluginDojo/eduPropertyEditor");
      }
    }
  }
}
}
```

3. Save and close the file.

Procedure 6: Validate your plug-in

In this procedure, you update the plug-in with the admin tool and validate it. You create a Product class document and verify that the custom property editor is available for the "Publisher" property.

1. In the Sample Desktop, verify that the "Publisher" property does not have a "Lookup" button before you implement the custom property editor.
2. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
3. In the Browse view, add a document to the `Sales` repository > `Manuals` folder.
   a. Click Add Document from the toolbar.
   b. For "What do you want to save", select "Information about a document" from the list.
   c. In the Properties section > Class field, select the `Book` class from the list and click OK.
   d. Select a value for the Document Title from the list.
4. Verify that the "Publisher" property does not have a "Lookup" button.



Properties

Class: Book

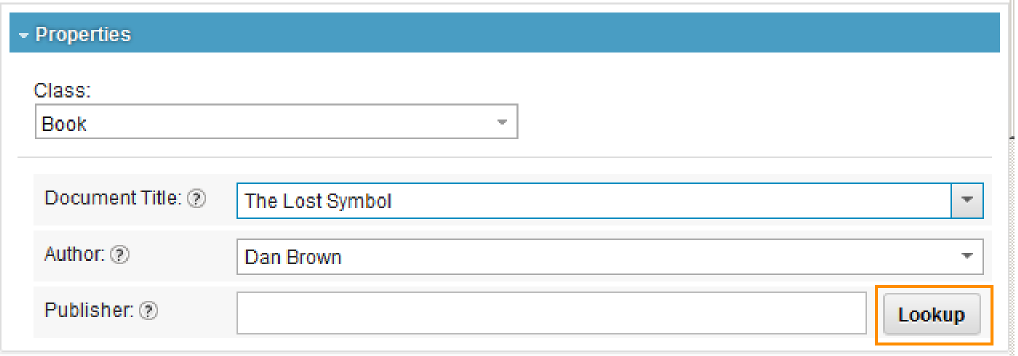
Document Title: The Lost Symbol

Author: Dan Brown

Publisher:

   e. Click Cancel.
5. Select the "Administration" view by clicking the "gear" icon in the navigation bar on the left side of the user interface.
6. In the Admin desktop, click the Plug-ins icon in the left pane.
7. In the Plug-ins tab, select the `EDU ICN Plugin` and click Edit.
   a. Make sure that the values are already filled for the "Class file path" and "Class name".
   b. Click Load to reload the plug-in with the changes.
   c. Click Save and Close.
8. Log out of Content Navigator and close the browser.
9. Verify that the "Publisher" property has a "Lookup" button after you implement the custom property editor.

10. Repeat steps 2 - 3 to add a document of the Product class.
11. Verify that the “Publisher” property has a button that is called “Lookup”.



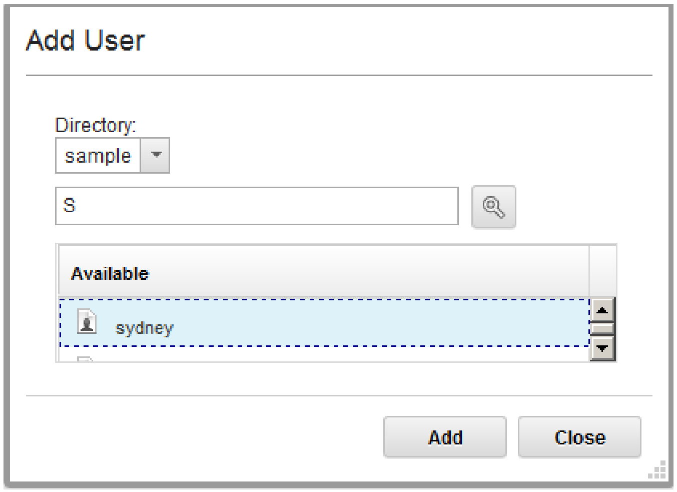
Class: Book

Document Title: The Lost Symbol

Author: Dan Brown

Publisher:

12. Click “Lookup” to select a user.
    a. In the Add user page, search for a user (Example: name that starts with S) and click Add.



Add User

Directory: sample

S

Available
sydney

    b. In the Add Document page, verify that the Publisher field has the value that you selected.
    c. Click Add to complete the Add Document wizard.

13. Log out of Content Navigator and close the browser.

Troubleshooting

Refer to the solution and verify that the code in your files is correct. Delete the plug-in and create a new one to make sure the plug-in reloads. Use Firebug to debug (refer to the appendix for more details).

LESSON 2.4: Develop a Plug-in to Add a Custom Feature

What this lesson is about

This lesson shows how to create a new Browse feature for IBM Content Navigator that provides users with a “Windows Explorer” type view into the Enterprise Content Management repository.

What you should be able to do

After completing this lesson, you should be able to:

- Develop a plug-in to add a custom Browse feature.

How you will check your progress?

- Hands on labs.

Add a feature to IBM Content Navigator interface

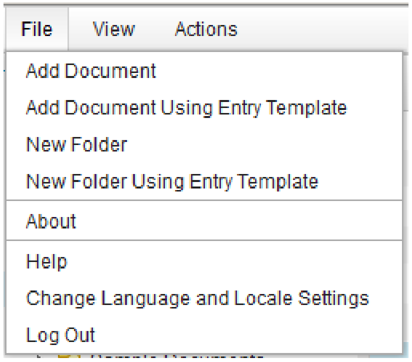
Introduction

An IBM Content Navigator feature refers to a top-level menu item that provides basic function in IBM Content Navigator such as Browse, Search, or Favorites. It also refers to one of the available extension points in plug-in development.

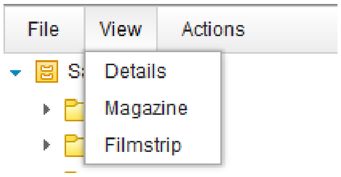
“Windows Explorer” type custom Browse view

In this lesson, you create a custom Browse feature that has a “Windows Explorer” type view and looks like the following screen captures.

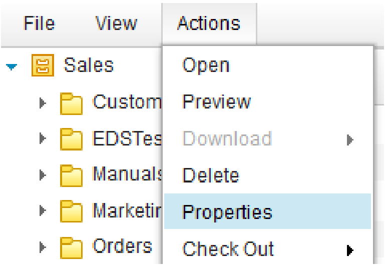
- You are able to access the common menu items from the File menu.


File
View
Actions
Add Document
Add Document Using Entry Template
New Folder
New Folder Using Entry Template
About
Help
Change Language and Locale Settings
Log Out

- The View menu enables you to browse the items in Details, Magazine, and Filmstrip views.



- You are able to access the common actions from the Actions menu.

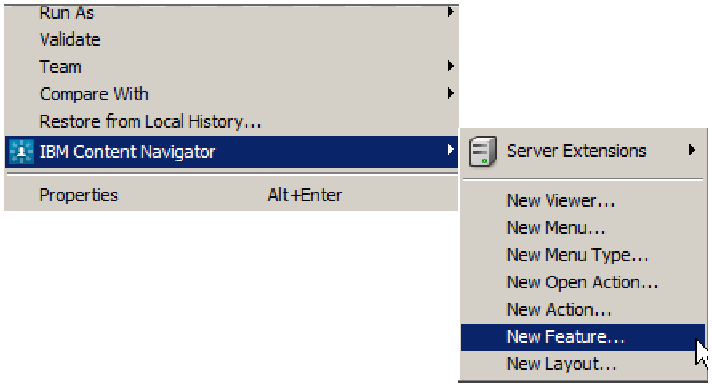


Creating client-side plug-in extensions for a feature

Creating a custom Feature

To create a new feature, you use the Eclipse plugin for IBM Content Navigator development wizard.

The screen capture shows how to add a Feature.



- Features are user interface panels that are exposed inside the IBM Content Navigator standard layout widget.
  - Example: “Search” and “Browse” panels in IBM Content Navigator.
- Plug-in developers can add custom features that can be entirely new capabilities to IBM Content Navigator or extend existing features.

Wizard-generated files for the new feature

The wizard generates the following files:

- EDUFeature.java
  - The Java class that defines the feature.
- EDUFeature.js
  - The Dojo module that implements the feature widget. The getContentClass() method references it in the EDUFeature.java file.
- EDUFeature.html
  - The template file that defines the layout of the feature with the template mechanism of Dojo.
  - The wizard provides an empty <div> tag.
  - The empty <div> tag is the starting point to design the layout of the feature.
  - Initially, it sets the style for the feature icon that you provide in the wizard.

Files that are used for this lesson

Java Files

- EDUPlugin.java
- EDUFeature.java

The following Java files enable the menus for the custom Browse feature in the IBM Content Navigator interface:

- EDUFolderContextMenu.java
  - Creates folders-specific menu items such as Open Folder, Create Folder and Remove From Folder
- EDUItemContextMenu.java
  - Creates menu items such as DefaultCheckOut, CheckIn, and DefaultSendEmail.
- EDUMixItemsContextMenu.java
  - Creates menu items such as SendLinkToSearches and Export.
- EDUSystemItemContextMenu.java
  - Creates menu items such as View, DeleteItem, and ShowHyperlink.

Note

All the menus that are listed here can be created in the IBM Content Navigator Administration tool. Refer to the “Customize a Desktop” unit > “Customize Menus” topic for details. The Java files demonstrate how you can configure a plug-in to create new menu types and toolbar types.

JavaScript Files

- EDUFeature.js
  - The Dojo module that implements the feature widget.
  - The getContentClass() method references it in the EDUFeature.java file.

Other Files

- EDUFeature.html
  - A template is an HTML file that provides the base layout for a Dojo widget.
- EDUPlugin.css
  - This file adds a set of styles that are required for the new feature.

Exercise 2.4.1: Develop a plug-in to add a feature

Introduction

In this exercise, you extend the existing IBM Content Navigator Browse feature and add the File menu bar, similar to the one in Windows Explorer. You also change the default menus to remove actions that belong to other features. Finally, you create a new desktop and enable the new feature and its menus.

Procedures

Procedure 1: Create a feature

Procedure 2: Import the Java classes

Procedure 3: Edit the EDUPlugin.java file

Procedure 4: Edit or import the Style Sheet

Procedure 5: Add the Browse widget and the HTML template

Procedure 6: Set up a new desktop to test the custom Browse feature

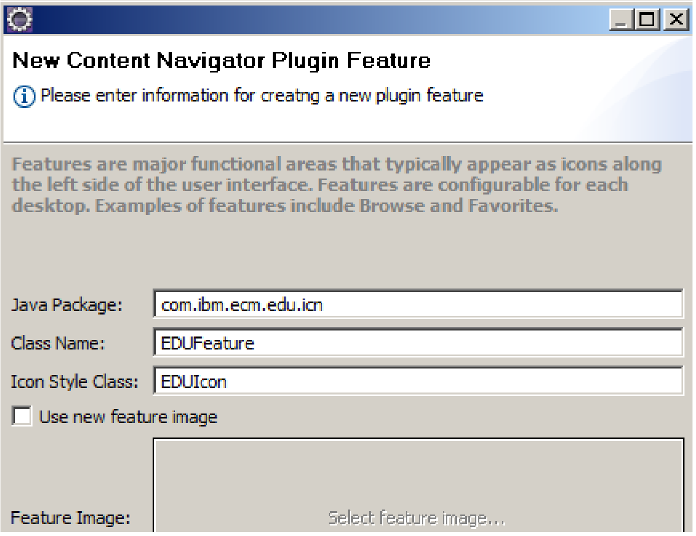
Procedure 1: Create a feature

In this procedure, you use the Eclipse plug-in for the IBM Content Navigator development wizard to create a new feature files.

1. Right-click the package (`com.ibm.ecm.edu.icn`) in your plug-in project and select the IBM Content Navigator > New Feature menu item.
2. In the New Content Navigator Plugin Feature page, the wizard fills your Java Package name automatically.
   a. Enter `EDUFeature` for the Class Name field.
   b. Enter `EDUIcon` for the Icon Style Class field.

Note

To select an image for the new feature, you must select Use new Feature image, which enables the Select feature image option. Click the option and go to the image location and select it. If configuration is correct, the icon is shown next to the Feature Image label. For this lab, an image is not required.



New Content Navigator Plugin Feature

Please enter information for creatng a new plugin feature

Features are major functional areas that typically appear as icons along the left side of the user interface. Features are configurable for each desktop. Examples of features include Browse and Favorites.

Java Package: com.ibm.ecm.edu.icn

Class Name: EDUFeature

Icon Style Class: EDUIcon

□ Use new feature image

Feature Image: Select feature image...

c. Click OK.

   The wizard generates the EDUFeature.java, EDUFeature.js, and EDUFeature.html files.

3. Check the wizard-generated code in the EDUFeature.java file.

   a. Verify that the `getContentClass()` method returns the Dojo module (`eDUPluginDojo.EDUFeature`).

4. You edit the EDUFeature.js and EDUFeature.html files in the following procedures.

Procedure 2: Import the Java classes

In this procedure, you import the new Java classes that define the customized versions of the existing IBM Content Navigator Item menus. These custom menus remove some actions that are not available in this simplified view of the repository.

1. Right-click the package (`com.ibm.ecm.edu.icn`) in your plug-in project and select Import from the menu.
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:\ICN\Plugins\Ex.4.4.1 folder and click OK.
3. Back in the Import > File system page, select the following files:
   - `EDUFolderContextMenu.java`
   - `EDUItemContextMenu.java`
   - `EDUMixItemsContextMenu.java`
   - `EDUSystemItemContextMenu.java`

   a. Make sure that the “`Into folder`” field has the following value: `EDUPlugin/src/com/ibm/ecm/edu/icn`
   b. Select the “Overwrite existing resources without warning” option.

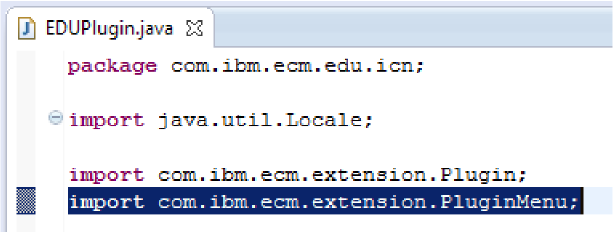
4. Click Finish.
5. Verify that the files are added in the templates package.
6. Open each file to see the menu options that they provide.

Procedure 3: Edit the EDUPlugin.java file

You must add a reference for the menu Java files that represent customized menus to the main Plugin java file. The EDUPlugin.java instructs IBM Content Navigator about their availability.

1. In Package Explorer, expand EDUPlugin > src > com.ibm.ecm.edu.icn.
   a. Double-click the `EDUPlugin.java` file.
2. Click the plus sign to expand the import at the top of the page and add the following package:

```
import com.ibm.ecm.extension.PluginMenu;
```



```
package com.ibm.ecm.edu.icn;

import java.util.Locale;

import com.ibm.ecm.extension.Plugin;
import com.ibm.ecm.extension.PluginMenu;
```

3. Verify that at the end of the file, a reference to EDUFeature exists.

   When you added EDUFeature to your project with the wizard, it updated that value automatically in the EDUPlugin.java file.
4. Add the code (after the EDUFeature entry) as shown in the following text. Bolded text indicates the required edits.

```
public com.ibm.ecm.extension.PluginFeature[] getFeatures(){
   return new com.ibm.ecm.extension.PluginFeature[] {
      new com.ibm.ecm.edu.icn.EDUFeature()};
}
public PluginMenu[] getMenus() {
   return new PluginMenu[] {
      new EDUFolderContextMenu(),
      new EDUItemContextMenu(),
      new EDUSystemItemContextMenu(),
      new EDUMixItemsContextMenu(),
   };
}
}
```

Note

The solution file (`EDUPlugin.java`) is available in the C:\ICN\Plugins\Ex.4.4.1 folder. Optionally, you can copy and paste the text to avoid typing errors or you can import the solution file to replace the existing one.

5. Save and close the file.

Procedure 4: Edit or import the Style Sheet

Note

Edit the `EDUPlugin.css` file to include styles for the custom feature. For this procedure, you can copy and paste the code from the solution to this file. To make it easy, you can import the solution file.

1. Right-click `com.ibm.ecm.edu.icn.WebContent` and 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:\ICN\Plugins\Ex.4.4.1 folder and click OK.
3. Back in the Import > File system page, select `EDUPlugin.css`.
   a. Make sure that the “`Into folder`” field has the following value: `EDUPlugin/src/com/ibm/ecm/edu/icn/WebContent`
   b. Make sure that you select the “Overwrite existing resources without warning” option.
4. Click Finish.
5. Expand the WebContent package and verify that the EDUPlugin.css file is added.
6. Open and check the file that it contains the styles for the EDUFeature.

Procedure 5: Add the Browse widget and the HTML template

The EDUFeature is an extension of the existing IBM Content Navigator BrowsePane. The extension is required to add the menu bar (similar to the one in Windows Explorer) and alter the custom modules that are applied to the ContentList widget.

In this procedure, you replace the wizard generated files with the solution files to add more code.

1. Import the files from the `C:\ICN\Plugins\Ex.4.4.1` folder into the packages with the data in the following table.

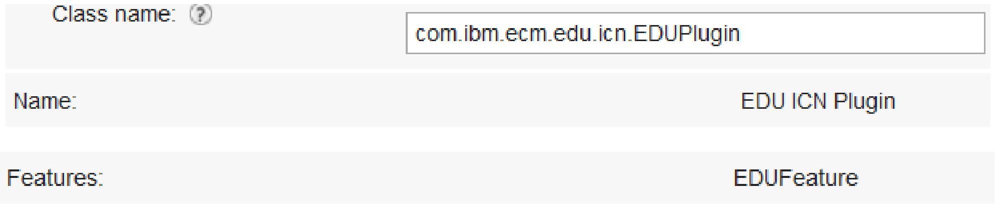
Package name in your project	Files to import
`com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo`	`EDUFeature.js`
`com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo.templates`	`EDUFeature.html`

2. Make sure that the "`Into folder`" field has the correct package.
   a. Select the "Overwrite existing resources without warning" option.
   b. Click Finish.
3. Verify that the files are listed in the Package Explorer.
4. Open the files and observe the code.

Procedure 6: Set up a new desktop to test the custom Browse feature

In this procedure, you define a new desktop with the custom Browse feature and test it.

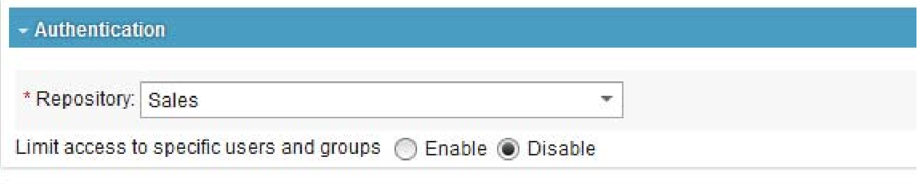
1. In your Firefox browser, go to `http://ecmedu01:9080/navigator/?desktop=admin`.
2. Enter the logon credentials for an administrator.
   - User ID: `p8admin`
   - Password: `IBMFileNetP8`
3. Update the plug-in.
   a. In the Admin desktop, click the Plug-ins icon in the left pane.
   b. In the Plug-ins tab, select the "`EDU ICN Plugin`" and click Edit.
   c. Make sure that the values are already filled for the "Class file path" and "Class name".
   d. Click Load to reload the plug-in with the changes.
   e. Verify that the details for the plug-in shows "Features: EDUFeature".



Class name:	com.ibm.ecm.edu.icn.EDUPlugin
Name:	EDU ICN Plugin
Features:	EDUFeature

   f. Click Save and Close.
4. Create a desktop.
   a. In the Admin desktop, select the Desktops tab and click New Desktop.
5. On the general tab, enter `EDU Feature Desktop` for the name field. The ID field is automatically filled.

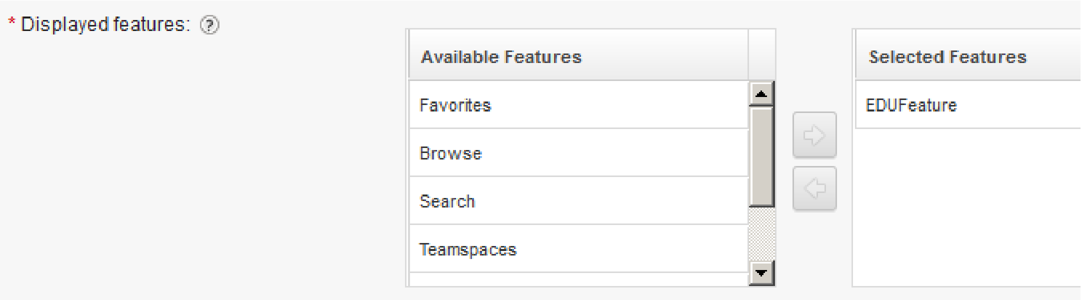
6. In the “Authentication” section, select `Sales` from the list for the Repository field.



* Repository: Sales

Limit access to specific users and groups ○ Enable ● Disable

7. In the Appearance tab, enter `EDU Feature Desktop` for the Application Name field.
   a. Click the down arrow to collapse the Banner and Login Page section.
8. In the Layout section > Selected Features pane, select all of the features and click the left arrow to remove them from the desktop.
   a. In the Available Features pane, select the EDUFeature and click the right arrow to add it to the desktop.

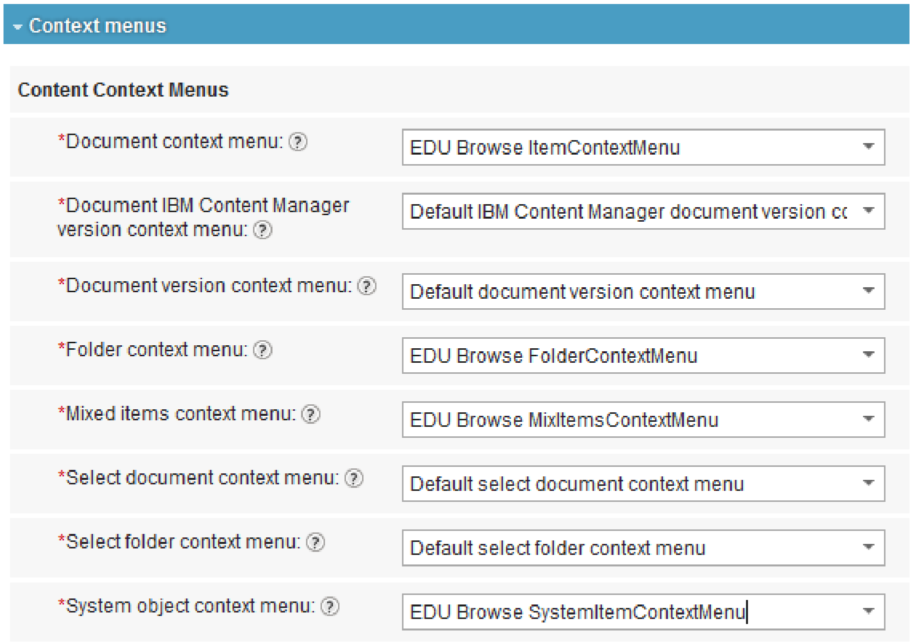


* Displayed features: ?

Available Features
Favorites
Browse
Search
Teamspaces

Selected Features
EDUFeature

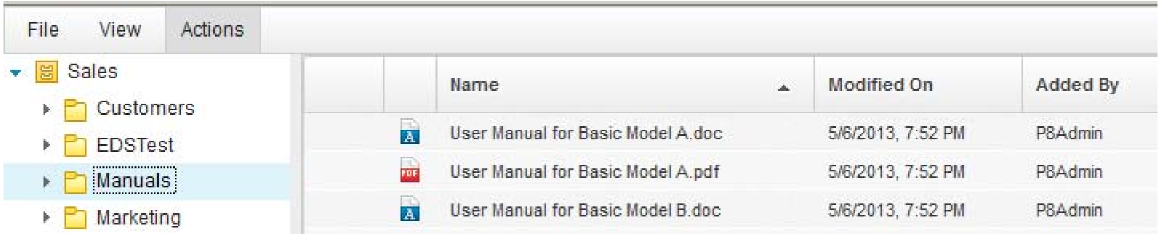
9. Click the Menus tab.
   a. Collapse the Toolbars section.
   b. Update the Document context menu, Folder context menu, Mixed items context menu, and the System object context menu to use the EDU Browse menu definitions instead of the default entries.



Content Context Menus

*Document context menu: ?	EDU Browse ItemContextMenu
*Document IBM Content Manager version context menu: ?	Default IBM Content Manager document version cc
*Document version context menu: ?	Default document version context menu
*Folder context menu: ?	EDU Browse FolderContextMenu
*Mixed items context menu: ?	EDU Browse MixItemsContextMenu
*Select document context menu: ?	Default select document context menu
*Select folder context menu: ?	Default select folder context menu
*System object context menu: ?	EDU Browse SystemItemContextMenu

10. Click Save and Close.
    a. Verify that the new Desktop is listed in the Desktops tab.
    b. Log out of IBM Content Navigator and close the browser.
11. Test your desktop.
    a. In your Firefox browser, log in to the new Desktop:
       – URL: `http://ecmedu01:9080/navigator/?desktop=EDUFeatureDesktop`
       – User ID: `p8admin`
       – Password: `IBMFileNetP8`
    b. Verify that the desktop opens with the custom Browse feature.
    c. Explore the different menus.



Name	Modified On	Added By
User Manual for Basic Model A.doc	5/6/2013, 7:52 PM	P8Admin
User Manual for Basic Model A.pdf	5/6/2013, 7:52 PM	P8Admin
User Manual for Basic Model B.doc	5/6/2013, 7:52 PM	P8Admin

12. Log out of EDU Feature Desktop. Click File > Log Out.
    a. Close the browser.

Troubleshooting

Refer to the solution and verify that the code in your files is correct. Delete the plug-in and create a new one to make sure the plug-in reloads. Use Firebug to debug (refer to the appendix for more details).

Lesson Summary

In this lesson, you completed the following tasks:

- Learned about developing a plug-in to add a custom feature for IBM Content Navigator.
- Created client-side extension to add the new feature.
- Added menu items through the plug-in.
- Validated your plug-in.

LESSON 2.5: Develop Plug-in to Add an Action

What this lesson is about

This lesson shows how to create a plug-in to add a new Action for IBM Content Navigator.

What you should be able to do

After completing this lesson, you should be able to:

- Create a plug-in to add a menu action.

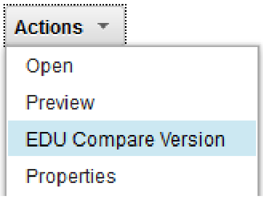
How you will check your progress?

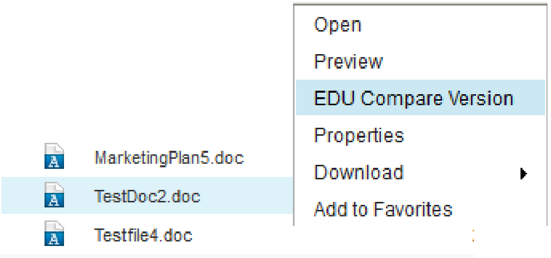
- Hands on Labs.

Develop a plug-in to add a menu action

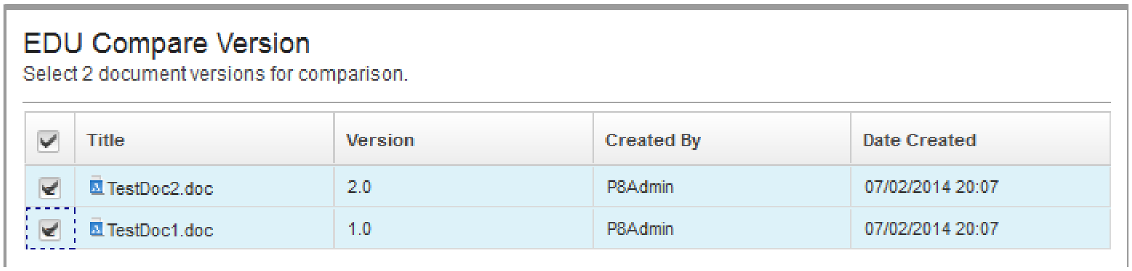
Create an EDU Compare Version action to the Document Content menu

In this lesson, you create a custom Action that is called EDU Compare Version and add it to the Document Context Menu.





- When you select a document with versions, and select the EDU Compare Version action for that document, the action retrieves the available versions for a document.



EDU Compare Version

Select 2 document versions for comparison.

☑	Title	Version	Created By	Date Created
☑	TestDoc2.doc	2.0	P8Admin	07/02/2014 20:07
☑	TestDoc1.doc	1.0	P8Admin	07/02/2014 20:07

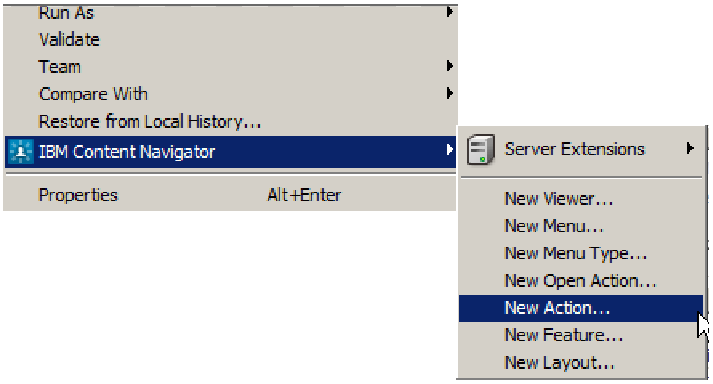
- You can select two versions of the document that you wanted to compare.

Creating client-side plug-in extensions for an action

Creating a custom action

- An action is an extension that allows the plug-in to define a new user interface action.
  - Typically actions show as buttons in a toolbar or menu items in a context menu. However, they can also be actions that are run implicitly during other user interactions within IBM Content Navigator.
  - You can create a plug-in to provide an action to use in IBM Content Navigator. This action can be added to toolbars and context menus.
- To create a new action, you use the Eclipse plug-in for IBM Content Navigator development wizard.

The screen capture shows how to add an Action.



Wizard-generated files for the new action

- The wizard generates a new Java class for the action, updates the primary plugin Java class, and updates the main plug-in JavaScript.
  - The changes to the plug-in Java class are necessary to instruct the IBM Content Navigator server that this plug-in includes a custom action.

The wizard generates the following files for this lesson:

- EDUCompareVersionAction.java
  - The Java class that defines the action.
- EDUPlugin.js
  - If the file exists, it adds code to the existing file.

Files that are used for this lesson

Java Files

- The following Java files enable the EDU Compare Version menu and the required service in IBM Content Navigator.
  - EDUPlugin.java
  - EDUCompareVersionAction.java
  - EDUCompareVersionService.java

JavaScript Files

- EDUPlugin.js
- EDUCompareVersionConfigPane.js
- The following files are the Dojo modules that implement the widgets for the action and service.
  - EDUCompareVersionActionModel.js
  - EDUCompareVersionDialog.js
  - EDUCompareVersionReport.js

Other Files

- The following files are the templates for the Dojo widgets.
  - EDUCompareVersionConfigPane.html
  - EDUCompareVersionDialog.html
  - EDUCompareVersionReport.html

Exercise 2.5.1: Develop a plug-in to add an action

Introduction

In this exercise, you create a plug-in for a custom action and add it to the “Document context menu”. You add the customized menu with the new action to a desktop and verify the action.

Procedures

Procedure 1: Create an action

Procedure 3: Import the Java classes

Procedure 4: Add the Dojo modules and the HTML templates for the action

Procedure 5: Edit the EDUPlugin.java file

Procedure 6: Update the plug-in

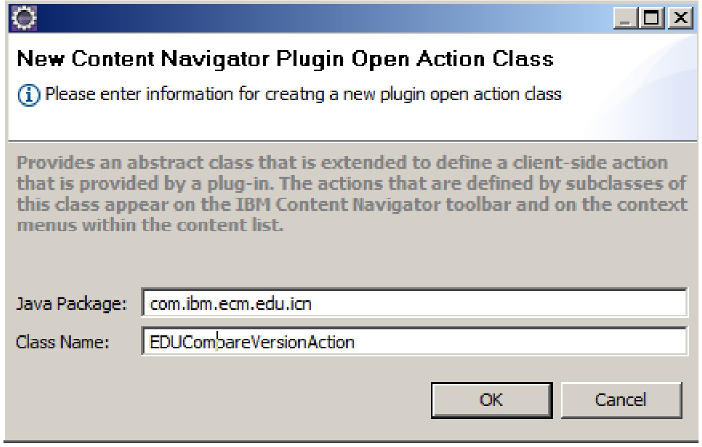
Procedure 7: Add the new action to a menu

Procedure 8: Assign the new menu to your desktop

Procedure 1: Create an action

In this procedure, you use the Eclipse Plug-in for IBM Content Navigator development wizard to create a new action files.

1. In Eclipse, right-click the package (`com.ibm.ecm.edu.icn`) in your plug-in project and select the IBM Content Navigator > New Action menu item.
2. In the New Content Navigator Plug-in Action page, the wizard fills your Java Package name automatically.
   a. Enter `EDUCompareVersionAction` for the Class Name field.



New Content Navigator Plugin Open Action Class

Please enter information for creatng a new plugin open action class

Provides an abstract class that is extended to define a client-side action that is provided by a plug-in. The actions that are defined by subclasses of this class appear on the IBM Content Navigator toolbar and on the context menus within the content list.

Java Package: com.ibm.ecm.edu.icn

Class Name: EDUCompareVersionAction

b. Click OK.

The wizard generates the EDUCompareVersionAction.java, and updates the EDUPlugin.js and EDUPlugin.java files.

Procedure 2: Add the required JAR files

In this procedure, you add JAR files that are required for this project.

1. Right-click your plug-in project (`EDUPlugin`) and select Properties from the menu.
2. In the Properties page, select Java Build Path from the left pane.
   a. Select the Libraries tab in the right pane.
   b. Click “Add External JARs”.
   c. In the JAR Selection page, go to C:\ICN\lib.
   d. Select the following files: (Press control and click the files to select multiple files).
      – Jace.jar
      – JSON4J_Apache.jar
   e. Click Open.
   f. Back in the Properties page > Libraries tab, verify that the files are added.
   g. Click OK.

Procedure 3: Import the Java classes

In this procedure, you import a new Java class that defines a service for running the action. You also import the action Java class to replace the wizard generated file to add more code.

1. Right-click the package (`com.ibm.ecm.edu.icn`) in your plug-in project and select Import from the menu.
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:\ICN\Plugins\Ex.4.5.1 folder and click OK.
3. Back in the Import > File system page, select the following files:
   – `EDUCompareVersionAction.java`
   – `EDUCompareVersionService.java`
   a. Make sure that the “`Into folder`” field has the following value: `EDUPlugin/src/com/ibm/ecm/edu/icn`
   b. Select the “Overwrite existing resources without warning” option.
4. Click Finish.
5. Verify that the files are added in the templates package.
6. Open each file to see the menu options that they provide.

Procedure 4: Add the Dojo modules and the HTML templates for the action

In this procedure, you import the files to add logic for the action, and for the service to run the action. You also import the templates that are needed for the Dojo modules.

1. Import the files from the `C:\ICN\Plugins\Ex.4.5.1` folder into the packages with the data in the following table.
2. Make sure that the "`Into folder`" field has the correct package.
   a. Select the "Overwrite existing resources without warning" option.

Package name in your project	Files to import
`com.ibm.ecm.edu.icn.WebContent`	`EDUPlugin.js`

`com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo`	`EDUCompareVersionActionModel.js`
	`EDUCompareVersionConfigPane.js`
	`EDUCompareVersionDialog.js`
	`EDUCompareVersionDialog2.js`
	`EDUCompareVersionReport.js`

`com.ibm.ecm.edu.icn.WebContent.eDUPluginDojo.templates`	`EDUCompareVersionConfigPane.html`
	`EDUCompareVersionDialog.html`
	`EDUCompareVersionReport.html`

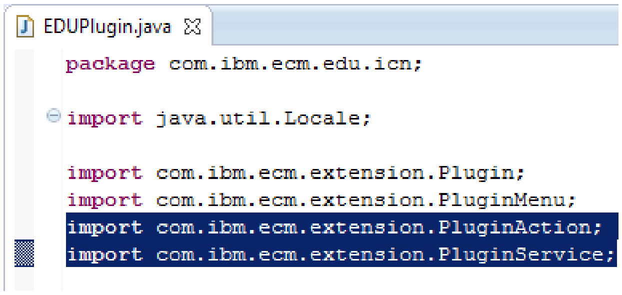
   b. Click Finish.
3. Verify that the files are listed in the Package Explorer.
4. Open the files and observe the code.

Procedure 5: Edit the EDUPlugin.java file

You must add a reference for the action and service Java files in the main plug-in Java file. The EDUPlugin.java instructs IBM Content Navigator about their availability.

1. In Package Explorer, expand EDUPlugin > src > com.ibm.ecm.edu.icn.
   a. Double-click the `EDUPlugin.java` file.
2. Click the plus sign (+) to expand the import at the top of the page and add the following packages:

```
import com.ibm.ecm.extension.PluginAction;
import com.ibm.ecm.extension.PluginService;
```



```
package com.ibm.ecm.edu.icn;

import java.util.Locale;

import com.ibm.ecm.extension.Plugin;
import com.ibm.ecm.extension.PluginMenu;
import com.ibm.ecm.extension.PluginAction;
import com.ibm.ecm.extension.PluginService;
```

3. Verify that at the end of the file, a reference to EDUCompareVersionAction exists.

   When you added EDUCompareVersionAction to your project with the wizard, it updated that value automatically in the EDUPlugin.java file.

4. You must add a reference manually to the service and ConfigurationDijitClass that you added in the earlier procedure.

   a. Add the code as shown in the following text. Bolded text indicates the required edits.

```
public PluginMenu[] getMenus() {
   return new PluginMenu[] {
      new EDUFolderContextMenu(),
      new EDUItemContextMenu(),
      new EDUSystemItemContextMenu(),
      new EDUMixItemsContextMenu(),
   };
}
public PluginService[] getServices() {
   return new PluginService[] {
      new EDUCompareVersionService()};
}
public String getConfigurationDijitClass() {
   return "eDUPluginDojo.EDUCompareVersionConfigPane";
}
```

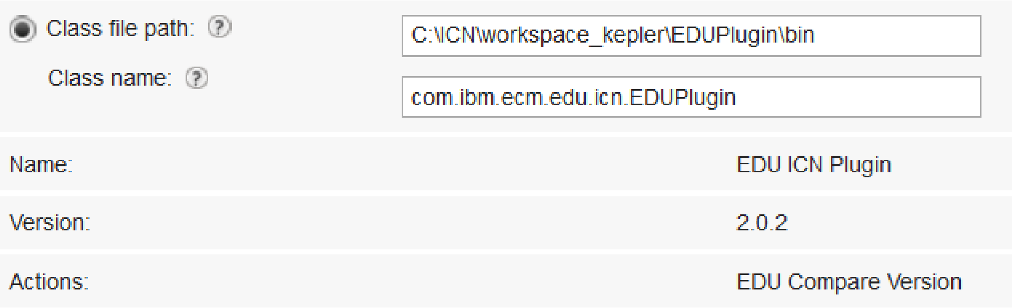
Note

The solution file (`EDUPlugin.java`) is available in the C:\ICN\Plugins\Ex.4.5.1 folder. Optionally, you can copy and paste the text to avoid typing errors or you can import the solution file to replace the existing one.

5. Save and close the file.

Procedure 6: Update the plug-in

1. In your Firefox browser, go to `http://ecmedu01:9080/navigator/?desktop=admin`.
2. Enter the logon credentials for an administrator.
   - User ID: `p8admin`
   - Password: `IBMFileNetP8`
3. Update the plug-in.
   a. In the Admin desktop, click the Plug-ins icon in the left pane.
   b. In the Plug-ins tab, select the `EDU ICN Plugin` and click Edit.
   c. Make sure that the values are already filled for the class file path and class name.
   d. Click Load to reload the plug-in with the changes.
   e. Verify that the details for the plug-in shows “Actions: EDU Compare Version”.



● Class file path: C:\ICN\workspace_kepler\EDUPlugin\bin

Class name: com.ibm.ecm.edu.icn.EDUPlugin

Name:	EDU ICN Plugin
Version:	2.0.2
Actions:	EDU Compare Version

   f. Click Save and Close.
   g. Refresh the browser.

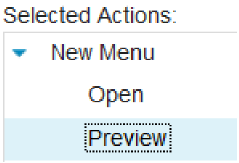
Procedure 7: Add the new action to a menu

After the plug-in is registered, the action that is associated with the plug-in is available for adding to a menu.

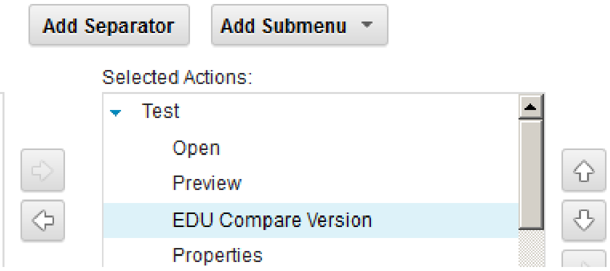
1. The Admin desktop is already opened.
   a. Expand the Menus node in the left pane.
   b. Find the “Default document context menu”.
2. The default menu cannot be modified. Make a copy: right-click and select Copy.

   This step creates a new menu of the same type.

3. Enter a name for the custom menu (Example: EDU Compare Version).
   a. The id is automatically populated based on the name you specify.
4. In the Selected Actions pane, select an item after which you want to show the custom action. (Example: Preview).



5. From the Available Actions on the left, select the “EDU Compare Version” action and move it to the Selected Actions.



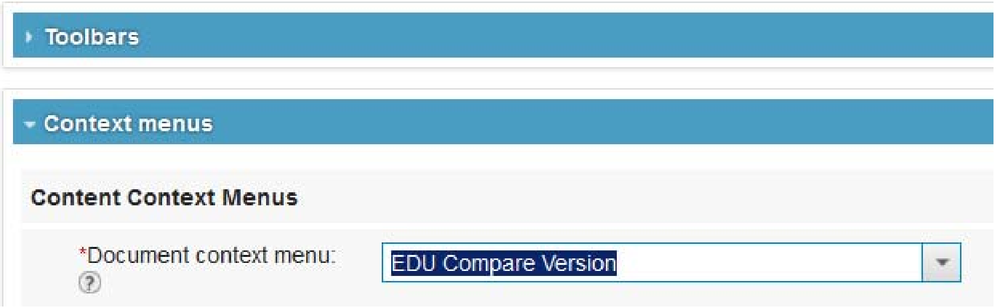
6. Click Save and Close.
7. Leave the admin desktop opened for the next procedure.

Procedure 8:Assign the new menu to your desktop

In this procedure, you replace the existing default “Document context menu” for this desktop with the new one.

1. The Admin desktop is already open.
   a. In the Desktops tab, select Sample Desktop.
   b. Click Edit.
2. Assign the new menu.
   a. Click the Menus tab.
   b. Collapse the Toolbars by clicking the down arrow.

c. In the “Context menus” section, select the new menu that you created (EDU Compare Version) from the list for the “Document context menu”.



*Document context menu: EDU Compare Version

3. Click Save and Close.
4. Log out of IBM Content Navigator and close the browser.

Procedure 9: Test the action

1. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In the Browse view, select the Sales repository > Marketing folder.
3. Select a document that has many document versions (Example: `Testfile4`).
4. Right-click the document and select the EDU Compare Version action from the list.
   a. Verify that the EDU Compare Version page opens with a list of versions available for the document.
   b. Select two versions and click OK.

Note

This lab shows the capability to add a custom action to IBM Content Navigator. All the code required to complete the version comparison is not added.

5. Log out of IBM Content Navigator and close the browser.

Troubleshooting

Refer to the solution and verify that the code in your files is correct. Delete the plug-in and create a new one to make sure the plug-in reloads. Use Firebug to debug (refer to the appendix for more details).

3

Create a Custom Workflow Step Processor

This unit provides guidance to create custom workflow step processors with the IBM Content Navigator toolkit.

LESSON 3.1: Create a Custom Step Processor

What this lesson is about?

This lesson describes how to create custom workflow step processors with the IBM Content Navigator toolkit. It provides background information about step processors and the step processors that are delivered with IBM Content Navigator.

What you should be able to do?

After completing this lesson, you should be able to:

- Create custom workflow step processors

How you will check your progress?

- Hands on Labs

Lab Solution files

- The solution files for this lesson are included in the following folder:
  C:\ICN\CustomStepProc

References

- IBM FileNet P8 5.2 Information Center
- IBM DeveloperWorks forums
- IBM Developer Works Technical Library
- IBM Redbooks publication: Customizing and Extending IBM Content Navigator

Workflows in IBM Content Navigator

Workflow overview

- A workflow is a series of work steps that are connected with routes.
- The routes determine which step is processed next based on the output of the previous step.
- The steps do the actual tasks of the workflow.
- Workflow steps can be categorized into work performers and step processors.

Work performers

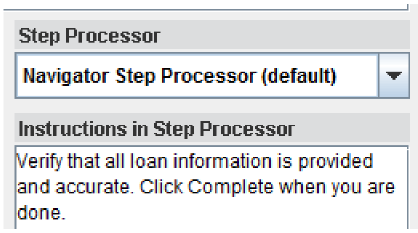
- A work performer is an automated task that contains no user interface.
- Work performers do the following actions that are based on the information in the workflow.
  - Update the attached documents
  - Call external systems
  - Perform IBM FileNet P8 operations

Step processors

- A step processor provides a user interface and is added to a workflow queue for processing.
  - When a user runs a workflow step, the user interface is shown.
  - The user finishes the appropriate actions and completes the step.
  - The workflow uses the routes from that step to determine the next step to run.

Workflow design

- Workflow Designers create and modify workflows in the Process Designer tool.
- When designing a workflow, add a step (Activity) to the workflow in the design space.
- Specify a step processor for the step through the properties user interface.
  - If IBM Content Navigator is installed and configured, the Navigator Step and Launch Processors are shown in the list of available step processors.
  - If any custom step processors are registered, they are also listed.



Step Processor

Navigator Step Processor (default)

Instructions in Step Processor

Verify that all loan information is provided and accurate. Click Complete when you are done.

- Configure the steps.

  The following customizations are available for a step:

  - Add Instructions to provide the information that the user needs to complete the task.
  - Select workflow properties and specify whether the properties are read-only or read/write.
  - Select attachments that must be shown to the user.
  - Set the following options:
    - The user who is processing the step can reassign the work to another user.
    - The History tab is shown.
    - The workflow status is shown.
- Link the step to other steps and complete the workflow design.

Workflow launch

You can launch a workflow manually or automatically with a Workflow Subscription.

Manual launch

- Select a workflow in the IBM Content Navigator Browse view and click Launch from the list of actions.
- Select a document for the workflow attachment.

Automatic launch with workflow subscription

- Associate a workflow with a Document class (Example: LoanApp) with Workflow Subscription.
  - When you add a document to the system and assigned to the LoanApp document class, an event action that is associated with the subscription is triggered that starts a Loan Processing workflow.
- Specify the attachment as an “Initiating Document” when you design the workflow.
  - That document that you add to the system is included as an attachment to the workflow.

Note

“IBM FileNet BPM 5.0: Process Design” course provides the training for designing workflows. “IBM FileNet BPM 5.0: Administration” course provides the training for managing the workflow system. Click here to find these trainings at the Enterprise Content Management Training path.

IBM Content Navigator step processors

Step processors overview

IBM Content Navigator provides two types of step processors that are configurable for workflows. They are widgets and they extend the ecm/widget/process/ProcessorLayout dijit. They can be used for many use cases:

1. Launch processor

It is a special type of step processor that is used as the first step in the workflow to start the workflow process. It can be used only as the first step in the workflow.

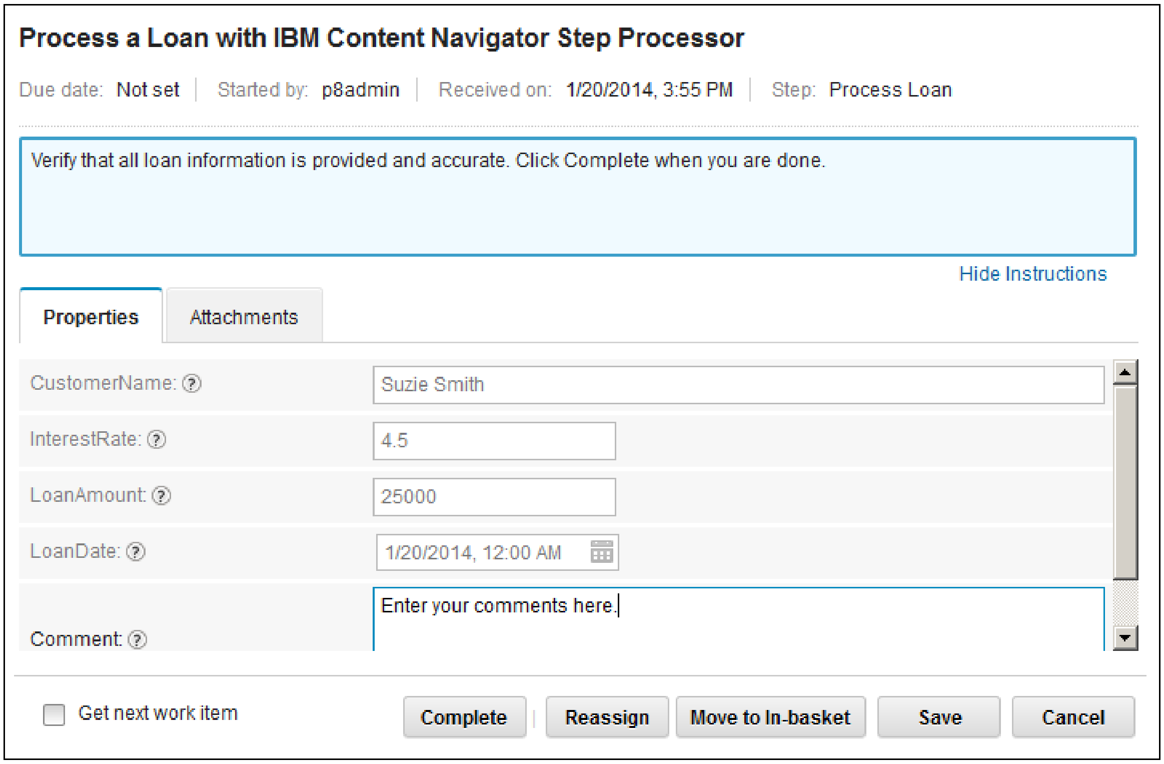
2. Step processor

The step processor can be used anywhere in a workflow except as the first step to provide a user interface for workflow steps that require user interaction.

Step processor user interface

The step processor has a multi-tab user interface that shows the information that is needed for the user to complete the step.

- The first tab shows the properties that are configured for the step.
  - The user can examine existing values and input values to complete the step.



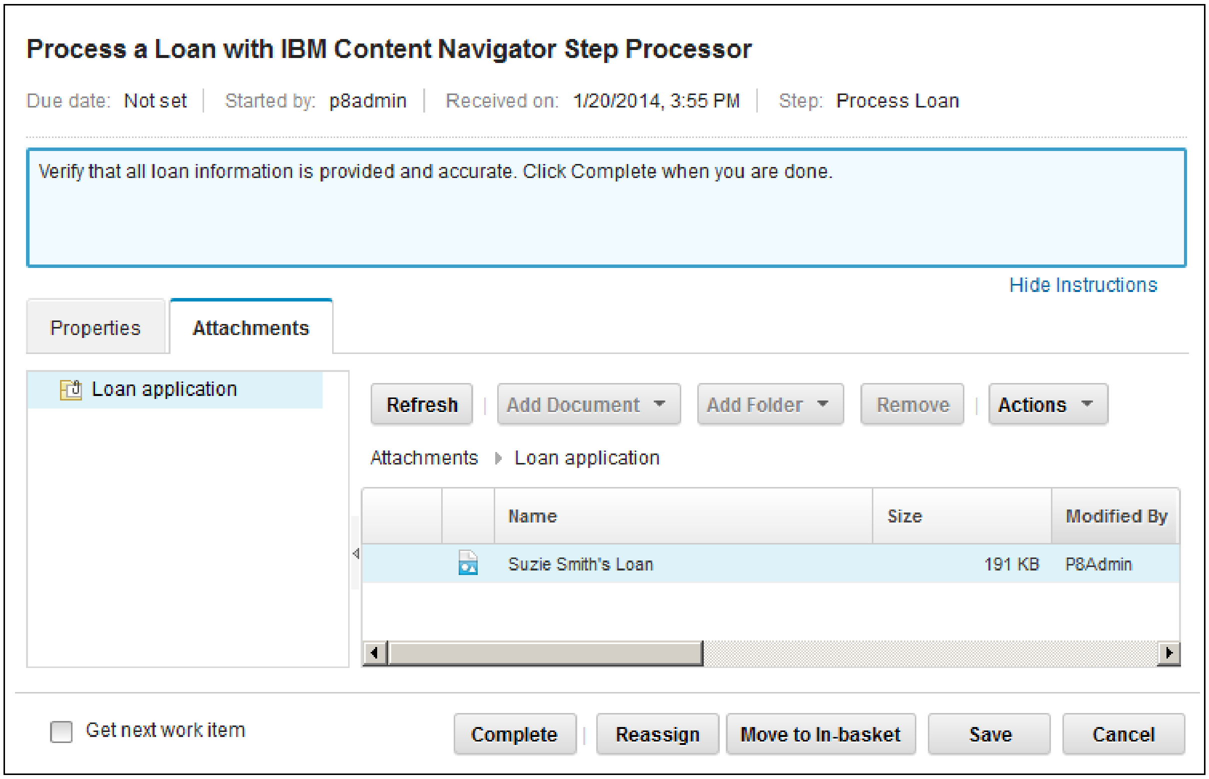
Process a Loan with IBM Content Navigator Step Processor

Due date: Not set | Started by: p8admin | Received on: 1/20/2014, 3:55 PM | Step: Process Loan

Verify that all loan information is provided and accurate. Click Complete when you are done.

CustomerName:	Suzie Smith
InterestRate:	4.5
LoanAmount:	25000
LoanDate:	1/20/2014, 12:00 AM
Comment:	Enter your comments here.

- The second tab shows any documents that are attached to the workflow and configured to display.
  - The user can add more documents and view the existing documents.



Process a Loan with IBM Content Navigator Step Processor

Due date: Not set | Started by: p8admin | Received on: 1/20/2014, 3:55 PM | Step: Process Loan

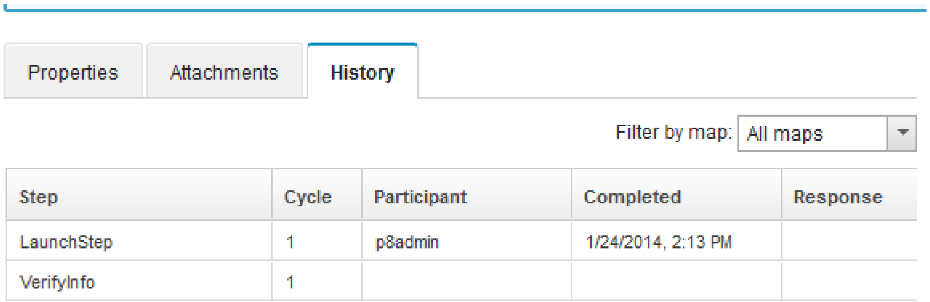
Verify that all loan information is provided and accurate. Click Complete when you are done.

Attachments ▸ Loan application

		Name	Size	Modified By
		Suzie Smith's Loan	191 KB	P8Admin

□ Get next work item

- The third tab (optional) shows the history of the workflow.



Filter by map: All maps

Step	Cycle	Participant	Completed	Response
LaunchStep	1	p8admin	1/24/2014, 2:13 PM	
VerifyInfo	1			

- These step processors provide buttons to do the following workflow step actions:
  - **Complete** the step
  - **Reassign** the work to someone else
  - **Move** the work item **to** the (personal) **In-basket**
  - **Save** the work in progress
  - **Cancel** the work item

Custom step processors

Custom step processors overview

The Content Navigator step processors are created with the Content Navigator toolkit. You can use the IBM Content Navigator step processor for many of the use cases for your work steps.

If you need more control of the step processor, you can customize it in the following ways.

- Create a custom step processor that extends the default Content Navigator step processor.
  - The custom step processor retains the features that the default step processor provides.
  - In addition, it includes more user interface components or functions.
- Modify the Content Navigator step processor user interface by adding, or modifying the widgets that are shown in the user interface.
- Integrate Content Navigator external data services for the properties that are shown in the custom step processor.

Existing Content Navigator Step Processor widget can be extended

When you extend the existing step processor widget, you need to write code only for the extensions that you want to provide.

- Following are the three files that implement the Content Navigator step processor:

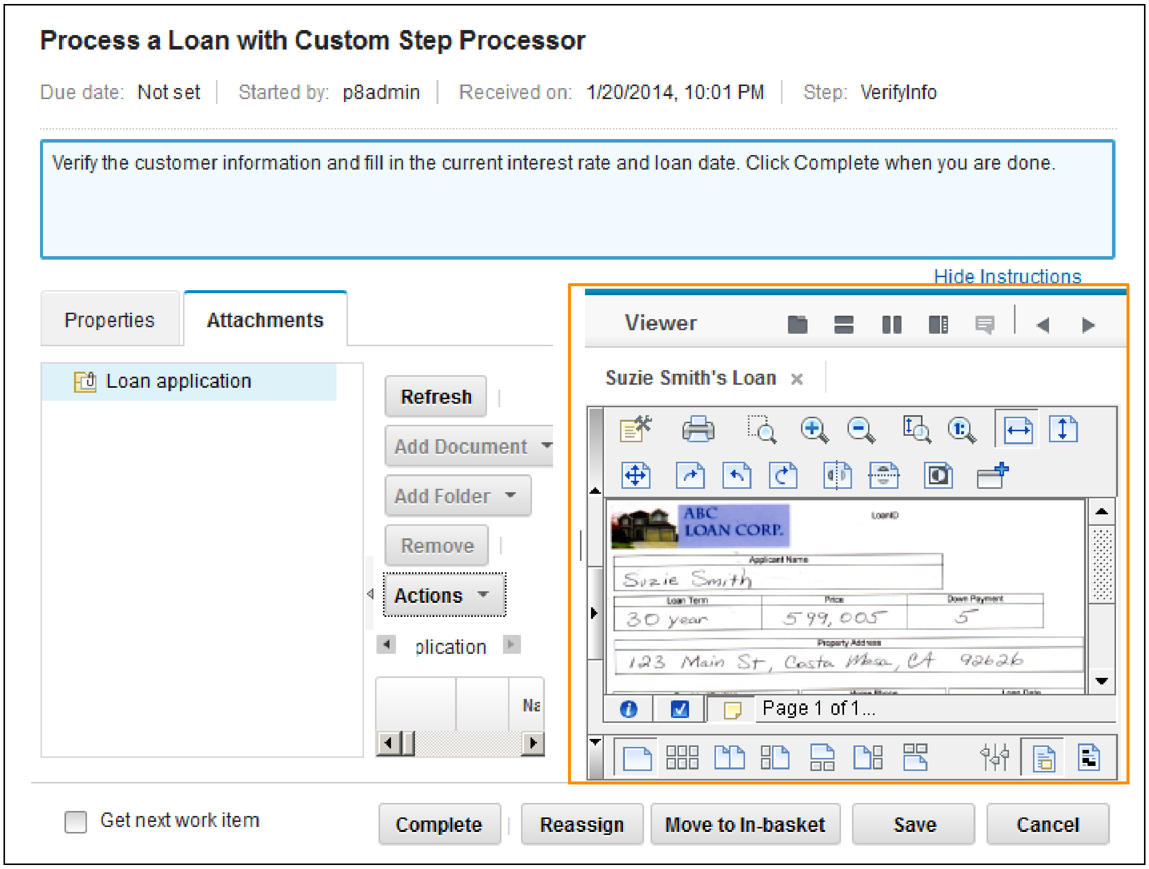
File name	Directory	Function of the file
stepprocessor.jsp	Content Navigator Deployment directory	Step processor controller. It is registered with the Process Configuration Console to identify it as a step processor that is used in workflows.
StepProcessorLayout.html	ecm\widget\process\templates	Provides the html template for the step processor widget.
StepProcessorLayout.js	ecm\widget\process	Creates the step processor widget.

Custom step processor that is created in this unit

In this lesson, you are going to do the following tasks in the lab:

- **Create a custom step processor for a workflow step**.
  - When you add a document and assign it to the LoanApp document class, an event action that is associated with the subscription is triggered. The event action starts a loan processing workflow.
  - This workflow uses a custom step processor to review the loan.

- **Add an embedded Content Navigator viewer.**
  - In the default Content Navigator step processor, you can view an attached document by opening the document in the attachments pane.
  - When you open the document, the Content Navigator viewer opens in a separate browser window to display the selected document.
  - The custom step processor embeds the Content Navigator viewer into the step processor user interface.
  - The viewer is in the same browser window as the step processor instead of in a separate browser window.



Process a Loan with Custom Step Processor

Due date: Not set | Started by: p8admin | Received on: 1/20/2014, 10:01 PM | Step: VerifyInfo

Verify the customer information and fill in the current interest rate and loan date. Click Complete when you are done.

- **Add an Action to use the embedded viewer.**
  - Add an Action that allows the document to be shown in the embedded viewer.
  - The default Content Navigator behavior is not overridden so the users have both the options: Embedded or non-embedded viewer.
- **External Data Services (EDS).**
  - Implement External Data Services to control the following properties that are shown in the step processor user interface.
    - Data validation for the loan number.
    - The dependent property choice lists for the Region and BranchOffice fields.

Demonstrations

Register a Custom Step Processor

Click here to watch the demonstration.

Assign a Custom Step Processor to a Workflow

Click here to watch the demonstration.

Modify a Workflow Subscription

Click here to watch the demonstration.

Exercise 3.1.1: Develop a custom step processor

Introduction

In this lab exercise, you create the files that are required for a custom step processor (that is extending the existing Content Navigator step processor). You deploy your custom step processor in the Content Navigator deployment directory.

Procedures

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

Procedure 2: Add a container for the embedded viewer in the custom processor

Procedure 3: Create a custom step processor widget

Procedure 4: Create a step processor controller

Procedure 5: Deploy the custom step processor

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: Add a container for the embedded viewer in the custom processor

In this procedure, you create the StepProcessorEDULayout.html file that provides the html template for the step processor widget. Copy and edit the existing html file to add a container for the viewer that is embedded in the custom step processor.

Note

In this lab exercise, the following directory is referred to as <`navigator deployment`> directory:
`C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\`
`navigator.ear\navigator.war`

1. Create a copy of the StepProcessorLayout.html delivered by Content Navigator from the `<navigator deployment>\ecm\widget\process\templates` directory.
2. Paste the file in the `C:\ICN\CustomStepProc\Source` directory.
3. Rename the file to `StepProcessorEDULayout.html` and open it in NotePad++ or in Eclipse for editing (shortcuts are on the desktop).
4. Modify the Step Content pane.

   In the default Content Navigator processor, the Step Content pane uses the full width of the step processor widget. You modify this pane so that the window is split between this pane and the Viewer pane that is added.

   a. Edit the existing `StepContentPane` (in line 21of the file) as shown in the following lines. The changes are shown in bold.

   ```
   <div data-dojo-type="dijit.layout.ContentPane"
   data-dojo-attach-point="stepContentPane" region="leading"
   splitter="true" style="width: 50%" >
   ```

   b. Change the region from "center" to "`leading`". This setting places the widget on the left.

   c. Add `splitter="true"` and `style="width: 50%"` attributes to this pane.

Note

This step adds a splitter between the stepContentPane and the Viewer pane and it specifies that the stepContentPane is allocated half of the width of the widget. The edits in Steps 3b and 3c change the layout to include the existing tab container for the step processor on the left and to add the viewer to the right side of the tab container.

5. Add the Viewer Container to the template.

   a. Add the following lines of code after the stepContentPane and before the ActionBar pane that is at the bottom of the widget. (Before the line 55: `<div data-dojo-type="dijit.layout.ContentPane" region="bottom">`)

   This Viewer Container holds the viewer widget that is added in the next procedure.

   ```
   <div data-dojo-attach-point="customViewerContainer"
   data-dojo-type="dijit.layout.ContentPane" region="center" style="width:
   50%" >
      <div id="contentViewer"style="width: 100%; height: 100%">
      </div>
   </div>
   ```

Hint

Optionally, copy the text from the `C:\ICN\CustomStepProc\Solution\StepProcessorEDULayout.html` file and paste to avoid any typing errors.

Procedure 3: Create a custom step processor widget

StepProcessorEDULayout creates the step processor widget. This widget extends the Content Navigator step processor so you are going to add the code only to include the viewer for the widget.

- – The existing widget that is extended provides the base function.
- – The widget is defined in the StepProcessorLayout.js file.

1. The `StepProcessorEDULayout.js` file is already created in your student system. Open this file in a text editor (Notepad++ or Eclipse) from the following directory and view the code.

   `C:\ICN\CustomStepProc\Source`
2. “`define`” section (line 22) defines the files that the widget requires.
3. “`declare`” section (line 30) declares the widget that you are creating.
4. The “`startup: function()`” initializes the viewer. This function is important.
   a. This method checks if the viewer is already created by checking contentViewer.
   b. If the viewer was not created already, then this procedure creates an instance of the Content Navigator viewer and assigns it to contentViewer.
5. Close the file without changes.

Procedure 4: Create a step processor controller

The stepprocessoredu.jsp file is the step processor controller.

- – This file is registered with the Process Configuration Console to identify it as a step processor that can be used in workflows.
- – This file is based on the stepprocessor.jsp file with only minor changes.
- – The customized controller uses the custom step processor widget instead of the step processor widget that is delivered with Content Navigator.

1. Create a copy of the `stepprocessor.jsp` delivered by Content Navigator from the `<navigator deployment>` directory.
   a. Paste the file in the `C:\ICN\CustomStepProc\Source` directory.
   b. Rename the file as `stepprocessoredu.jsp`.
   c. Open it in NotePad++ or Eclipse for editing.
2. Edit the file as shown in the following lines.

Hint

Optionally, copy the text from the `C:\ICN\CustomStepProc\Solution\stepprocessoredu.jsp` file and paste to avoid any typing errors.

3. The changes are shown in bold.
   a. Add a `<script>` block with `dojo.require` for the custom widget so that it can be used.
   b. Replace the existing controller to use the custom step processor. (Edit the line that contains `<div dojoType="ecm.widget.process.StepProcessorLayout"`)

```
      <div id="ECMWebUIloadingText" class="contentNode loadingText"></div>
    </div>
  <script>
      dojo.require("custom.widget.process.StepProcessorEDULayout");
  </script>
  <div dojoType="custom.widget.process.StepProcessorEDULayout"
  id="ECMStepProcUI" style="width: 100%; height: 100%"
  lang="<%=htmlLocale%>"></div>
</body>
</html>
```

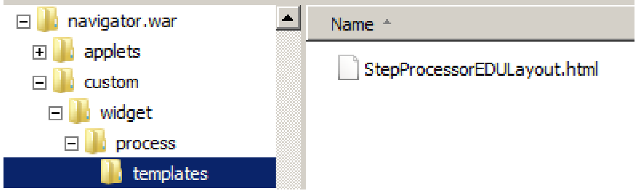
Procedure 5: Deploy the custom step processor

The implementation files must be installed directly into the Content Navigator deployed directories. In this procedure, you deploy the custom step processor.

1. Stop the Content Navigator application.
   a. In a Firefox browser, open the WebSphere Application Server Administration Console.
      – Click the WAS Admin link in the Bookmarks or go to the following URL: `https://ecmedu01:9043/ibm/console/logon.jsp`
   b. Enter the account information and click Login.
      – User ID: p8admin
      – Password: IBMFileNetP8
   c. In the left pane, expand Applications > Applications Types.
   d. Click the WebSphere enterprise applications link.
   e. Select `navigator` in the Enterprise Applications page.
   f. Click Stop and wait for the stop message to display.

2. In Windows Explorer, copy the custom step processor files into the Content Navigator deployment location on the application server.
   a. Copy `stepprocessoredu.jsp` into the `<navigator deployment>` directory.
   b. Under the `<navigator deployment>` directory, create four subfolders: `\custom\widget\process\templates`
   c. Copy `StepProcessorEDULayout.js` into the `process` directory that you created.

d. Copy `StepProcessEDULayout.html` into the `templates` directory.



3. Start the Content Navigator application in the WebSphere Application Server Administration Console.

   a. Select navigator in the Enterprise Applications page and click Start.

   b. Wait for the Start message to display.

4. Logout of the Administration Console and close the browser.

Note

You are going to test the custom step processor after all the configuration exercises are complete.

Exercise 3.1.2: Register the custom step processor

Introduction

The custom step processor must be registered through the Process Configuration Console before it can be used in a workflow. After registration:

- It becomes available in the Process Designer as one of the supported step processors.
- The FileNet P8 repository is able to find it when a step is executed.

Procedures

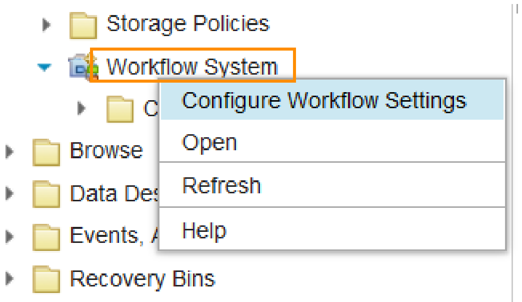
Procedure 1: Open the Process Configuration Console

Procedure 2: Troubleshooting the Java error

Procedure 3: Register the custom step processor

Procedure 1: Open the Process Configuration Console

1. Log in to the Administration Console for Content Platform Engine (ACCE) application.
   a. In a Firefox browser, go to http://ecmedu01:9080/acce
   b. Enter the login credentials.
      - User name: `P8admin`
      - Password: `IBMFileNetP8`
2. Start the Process Configuration Console.
   a. In the P8Domain tab, expand P8 Domain > Object Stores.
   b. Click LoanProcess.
   c. In the LoanProcess tab, expand LoanProcess > Administrative > Workflow System.
   d. Right-click Workflow System and click Configure Workflow Settings from the list.

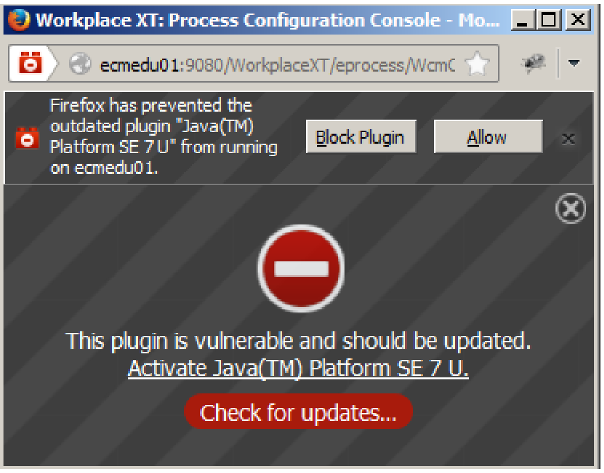


   e. The Process Configuration Console applet opens.

Procedure 2: Troubleshooting the Java error

Troubleshooting

If you get an error similar to the one in the following screen capture, when you try to open the Process Configuration Console or any other applet, do the following steps. Otherwise, skip to the next procedure.


Workplace XT: Process Configuration Console - Mo...
ecmedu01:9080/WorkplaceXT/eprocess/WcmC
Block Plugin
Allow
Check for updates...

Firefox has prevented the outdated plugin "Java(TM) Platform SE 7 U" from running on ecmedu01.

This plugin is vulnerable and should be updated.
Activate Java(TM) Platform SE 7 U.

1. Click the red icon at the top left corner.
   a. Click “Allow and Remember” and close the error window by clicking X (do not close the Process Configuration Console window).
   b. Do not update to new Java version as it might not work with this applet.

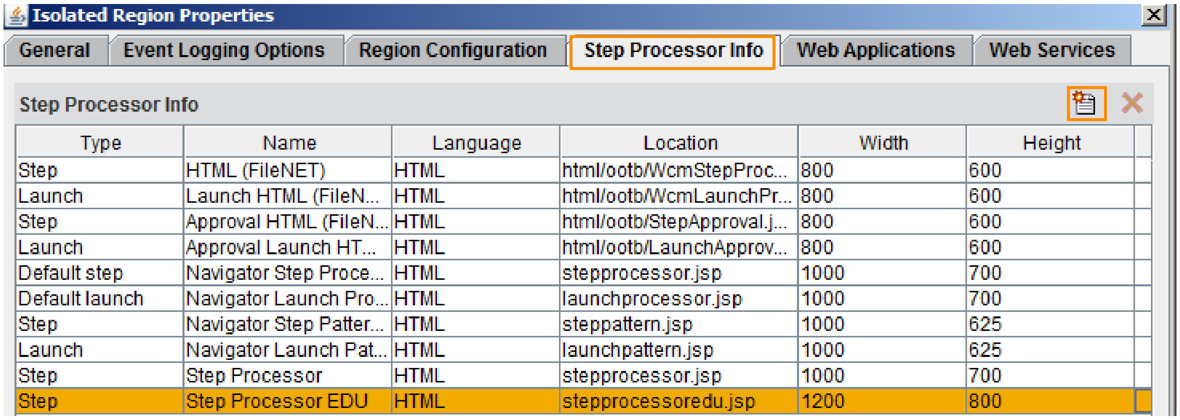
Procedure 3: Register the custom step processor

1. In the Process Configuration Console, right-click P8ConnP3[3] in the tree view on the left pane.
   a. Select Connect. The node expands and shows the folders under it.
   b. After the connection, right-click P8ConnP3[3] again and select Properties.
2. Enter Step Processor Information:
   a. In the Isolated Region Properties window, select the Step Processor Info tab.
   b. Click the Add icon in the upper right and a new row is created.
   c. Enter the information that is provided in the following table.

d. Double-click in the cell to edit the Name field

Type	Name	Language	Location	Width	Height
Step	Step Processor EDU	HTML	<see the steps below>	1200	800

To support a larger window for the embedded viewer, width 1200 and height 800 are used.

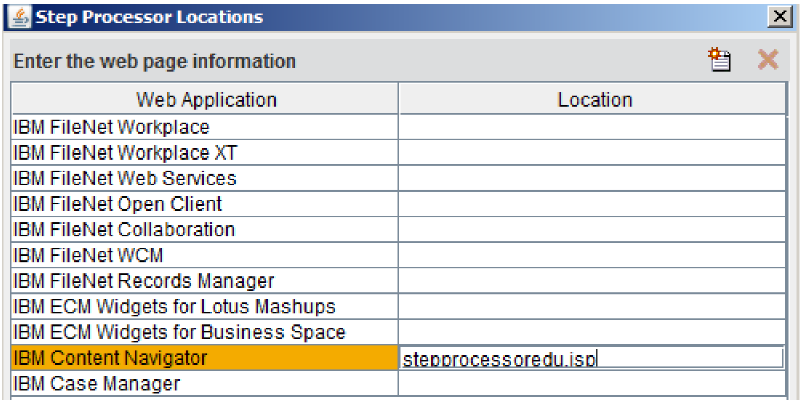


Step Processor Info

Type	Name	Language	Location	Width	Height
Step	HTML (FileNET)	HTML	html/ootb/WcmStepProc...	800	600
Launch	Launch HTML (FileN...	HTML	html/ootb/WcmLaunchPr...	800	600
Step	Approval HTML (FileN...	HTML	html/ootb/StepApproval.j...	800	600
Launch	Approval Launch HT...	HTML	html/ootb/LaunchApprov...	800	600
Default step	Navigator Step Proce...	HTML	stepprocessor.jsp	1000	700
Default launch	Navigator Launch Pro...	HTML	launchprocessor.jsp	1000	700
Step	Navigator Step Patter...	HTML	steppattern.jsp	1000	625
Launch	Navigator Launch Pat...	HTML	launchpattern.jsp	1000	625
Step	Step Processor	HTML	stepprocessor.jsp	1000	700
Step	Step Processor EDU	HTML	stepprocessoredu.jsp	1200	800

3. Enter the Step Processor location:

a. Double-click in the Location field. The Step Processor Locations window opens.



Enter the web page information

Web Application	Location
IBM FileNet Workplace	
IBM FileNet Workplace XT	
IBM FileNet Web Services	
IBM FileNet Open Client	
IBM FileNet Collaboration	
IBM FileNet WCM	
IBM FileNet Records Manager	
IBM ECM Widgets for Lotus Mashups	
IBM ECM Widgets for Business Space	
IBM Content Navigator	stepprocessoredu.jsp
IBM Case Manager	

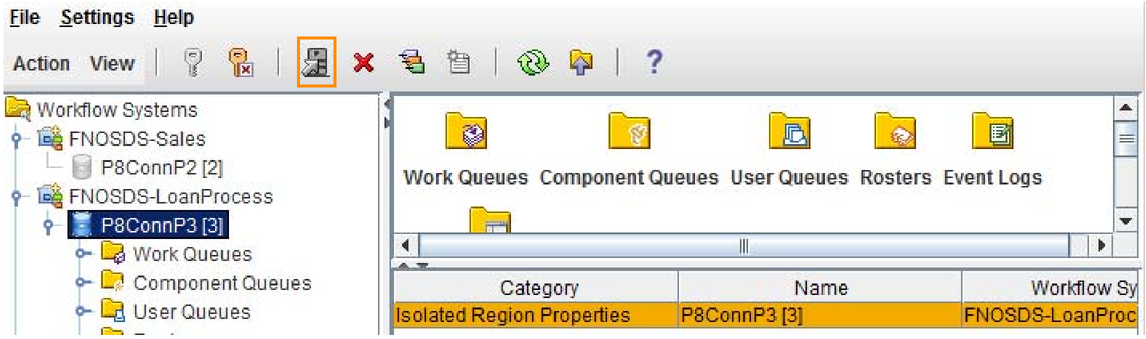
b. Select the IBM Content Navigator entry.

c. Enter `stepprocessoredu.jsp` in the Location field.

d. Click OK to close the Step Processor Locations window.

e. Click OK again to close the Isolated Region Properties window.

4. Commit the changes.
   a. In the main window, click the Commit Changes icon.



Category	Name	Workflow Sy
Isolated Region Properties	P8ConnP3 [3]	FNOSDS-LoanProc

5. In the Process Configuration Console window, click Continue.
6. When the window shows Success, click Close.
7. Leave the Process Configuration Console tool open for the next procedure.

Exercise 3.1.3: Configure Application Space Roles and In-baskets

Introduction

Content Navigator uses Application Space and in-baskets for showing a list of work items to be processed. Your student system is already configured with in-baskets for this lab. In this lab exercise, you verify the configuration.

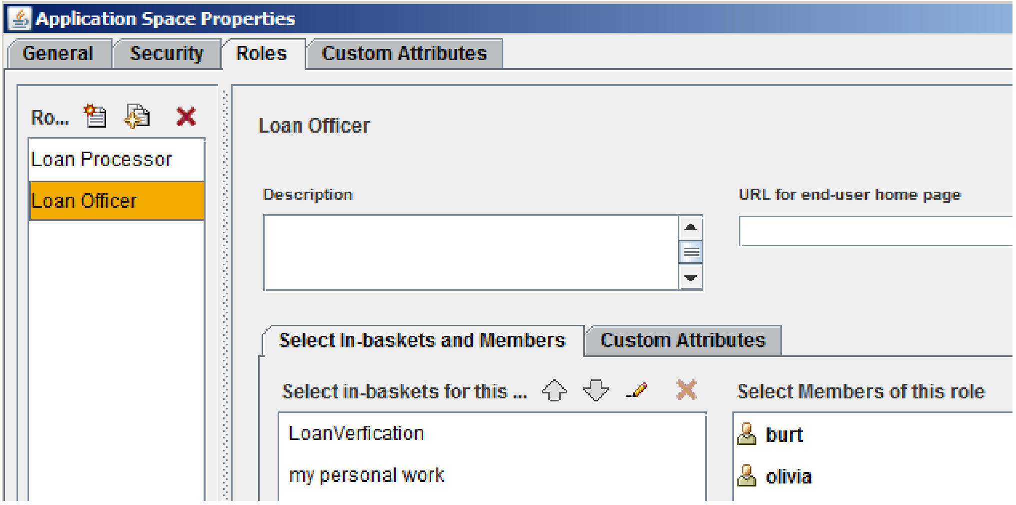
Procedures

Procedure 1: Verify the Application Space Roles configuration

Procedure 2: Verify the In-baskets configuration

Procedure 1: Verify the Application Space Roles configuration

1. You are already logged in and the Process Configuration Console tool is opened from the previous exercise.
1. In the Process Configuration Console tree view on the left, expand the P8ConnP3[3] > Application Spaces.
   a. Right-click DefaultApplication and select Properties.
   b. In the Application Space Properties window, click the Roles tab.
2. Verify that there are two Roles:
   – Loan Processor
   – Loan Officer
3. Select the Loan Officer role from the left pane.
4. Notice that on the right pane, a list of in-baskets and members are assigned to this role.


Application Space Properties
General
Security
Roles
Custom Attributes
Ro...
Description
URL for end-user home page
Select In-baskets and Members
Custom Attributes
Select in-baskets for this ...
Select Members of this role

Loan Processor

Loan Officer

Loan Officer

LoanVerfication

my personal work

burt

olivia

5. Examine the Loan Processor role.

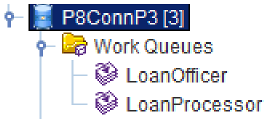
Note

In this user interface, you can create Roles for this application space and assign members and in-baskets.

c. Click OK to close the Application Space Properties window.

Procedure 2: Verify the In-baskets configuration

1. In the Process Configuration Console > tree view on the left, expand the P8ConnP3[3] > Work Queues.



a. Verify that there are two Work Queues:

– LoanProcessor

– LoanOfficer

For convenience, the roles and work queues are labeled with the similar names.

b. Right-click LoanOfficer and select Properties.

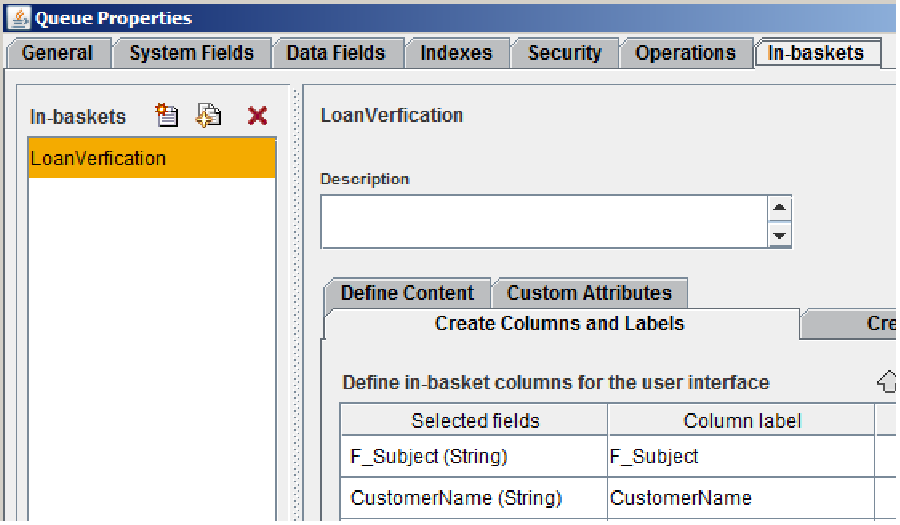
c. In the Queue Properties window, check that the LoanVerification in-basket.

Note

In this user interface, you configure in-baskets and then they are available in the Application Space.

d. Notice that on the right pane > Create Columns and Labels tab, a list of fields and column labels are defined.

e. The column labels are shown in the Content Navigator Work view when a work item is opened in the step processor.



LoanVerfication

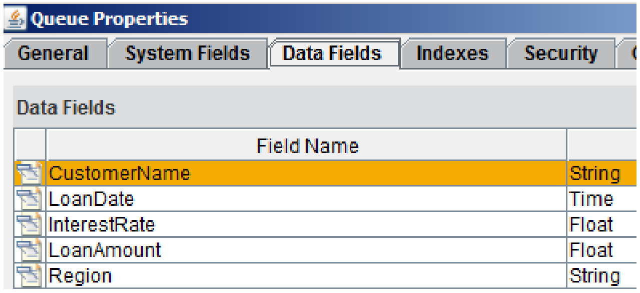
Description

Define in-basket columns for the user interface

Selected fields	Column label
F_Subject (String)	F_Subject
CustomerName (String)	CustomerName

f. On the Queue Properties page, click Data Fields tab and check the list of data fields that are defined for this queue.

g. Click OK to close the Queue Properties window.



Data Fields

Field Name	
CustomerName	String
LoanDate	Time
InterestRate	Float
LoanAmount	Float
Region	String

2. Optionally check the LoanProcessor Queue properties.
3. Exit the Process Configuration Console tool by clicking File > Exit.
4. Log out of the Administration Console for Content Platform Engine application.

Exercise 3.1.4: Assign the custom step processor to a workflow

Introduction

In the previous labs, you deployed a custom step processor and registered it. The step processor is now available in the Process Designer as one of the supported step processors. A workflow is already designed for this lesson.In this exercise, you modify the workflow to assign the custom step processor to a workflow step.

Procedures

Procedure 1: Open the Process Designer tool

Procedure 2: Assign the custom step processor to a workflow step

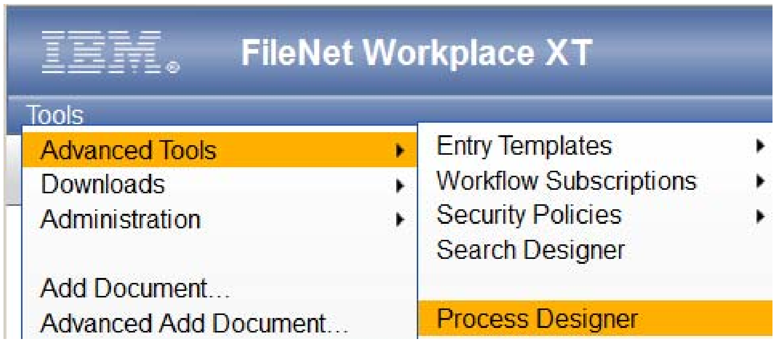
Procedure 1: Open the Process Designer tool

1. Log in to the FileNet Workplace XT application.
   a. In a Firefox browser, go to http://ecmedu01:9080/WorkplaceXT
   b. Enter the login credentials.
      – User name: `P8admin`
      – Password: `IBMFileNetP8`

Troubleshooting

If you are unable to access the Workplace XT, clear the cache in your Firefox browser. From the Tools menu, click the Clear Recent History. In the Clear All History window, click Clear Now.

2. Start the Process Designer tool.
   a. Click the Tools > Advanced Tools > Process Designer link from top menu.



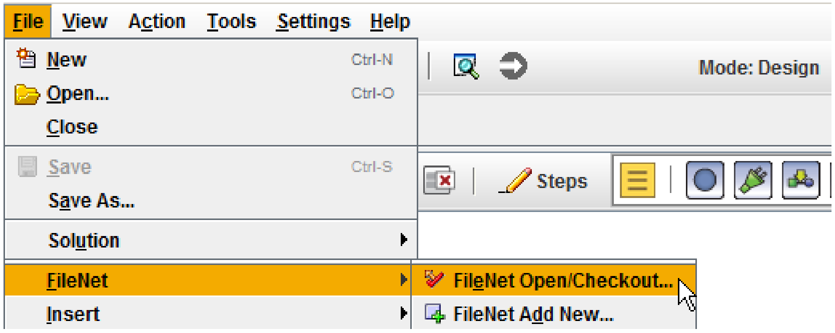
   b. The Process Designer applet opens.

Troubleshooting

When you open the Process Designer, if you get a Java error, do the steps in Procedure 2: Troubleshooting the Java error, page. 3.16.

Procedure 2: Assign the custom step processor to a workflow step

1. In Process Designer, click File > FileNet > FileNet Open/Checkout.



   a. In the “Open a Workflow Definition” window, select the LoanProcess object store > `Workflows` folder > ProcessLoan workflow.

   b. Select the `Checkout` option for “Open As”.

   c. Click Open.

   d. The workflow is opened in the designer window.

   e. This workflow contains a launch step and two activity steps.

2. Explore the Workflow properties:

   a. In the lower pane > Workflow Properties tab > General subtab, verify that Workflow name is ProcessLoan and a default value is provided for the Subject.

      You can edit the Subject when you are launching the workflow.

   b. Click the Data Fields subtab. A list of data fields are defined for this workflow.

   c. Click the Attachments subtab. An attachment array is defined.

      This configuration allows you to add documents as an attachment for this workflow.

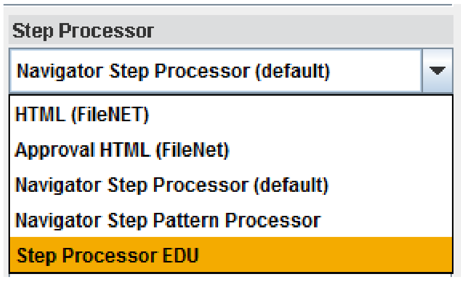
3. Assign the custom step processor to the VerifyInfo step:

   a. Click the VerifyInfo activity icon in the design space.

   b. In the lower pane > VerifyInfo tab > General subtab, verify that the step name and activity type (Work Queue > LoanOfficer) are defined.

   c. Notice that participants have Reassign, View status and View history privileges.

d. Under the Step Processor section, select your custom step processor from the list (Step Processor EDU).


Step Processor
Navigator Step Processor (default)
HTML (FileNET)
Approval HTML (FileNet)
Navigator Step Processor (default)
Navigator Step Pattern Processor
Step Processor EDU

e. In the Parameters subtab, check the data fields that are assigned for this step.

4. Validate the workflow:
   a. Click File > Validate Workflow Collection.
   b. Click Close when prompted with the message that the workflow validation is successful.
5. Transfer the workflow:
   a. Click File > Transfer Workflow Collection.
   b. In the “Checkin Workflow Definition” window, click Finish.
   c. If prompted, in the “Check workflow name” window, leave the default value “Use this workflow name”, and click OK.
   d. Click Close when prompted with the message that the Transfer is successful.
6. Close the Process Designer tool:
   a. Click File > Exit.
   b. If prompted, select “Cancel the checkout” option in the Process Designer window and click OK.
7. Log out of the Workplace XT application and close the browser.

Exercise 3.1.5: Modify the Workflow Subscription

Introduction

To start a workflow automatically, you must associate it with an object class through a Workflow Subscription. When you add a document and assign it to that Document class, an event action that is associated with the subscription is triggered that starts the workflow.

Specify the attachment as an “Initiating Document” when you design the workflow. The document that you add to the system is included as an attachment in the workflow.

A Workflow Subscription for this lab is already created. In this exercise, you modify the Workflow Subscription (that is associated with the LoanApp Document class) to update the ProcessLoan workflow version that you edited in the previous lab.

Procedures

Procedure 1: Open the Workflow Subscription

Procedure 2: Modify the Workflow Subscription

Procedure 3: Test the custom step processor

Procedure 1: Open the Workflow Subscription

1. Log in to the Administration Console for Content Platform Engine application.
   a. In a Firefox browser, go to http://ecmedu01:9080/acce
   b. Enter the login credentials.
      – User name: `P8admin`
      – Password: `IBMFileNetP8`
2. Open the Workflow Subscription.
   a. In the P8Domain tab, expand P8 Domain > Object Stores.
   b. Click LoanProcess.
   c. In the LoanProcess tab, expand the LoanProcess > Events, Actions, Processes > Subscriptions folder in the left pane.
   d. Click `CustomStepProcSubscription`.
   e. The CustomStepProcSubscription tab opens in the right pane.

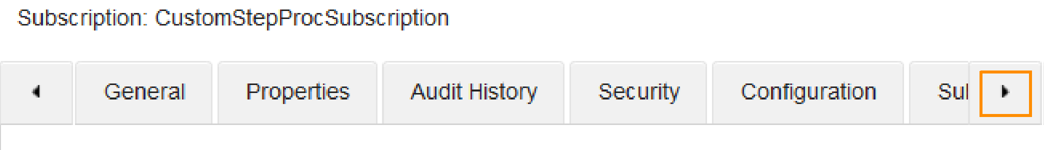
Procedure 2: Modify the Workflow Subscription

1. In the CustomStepProcSubscription tab > General subtab, notice that this subscription is associated with LoanApp class.

   The Target ID field has the class value.

2. Select the Workflow version:

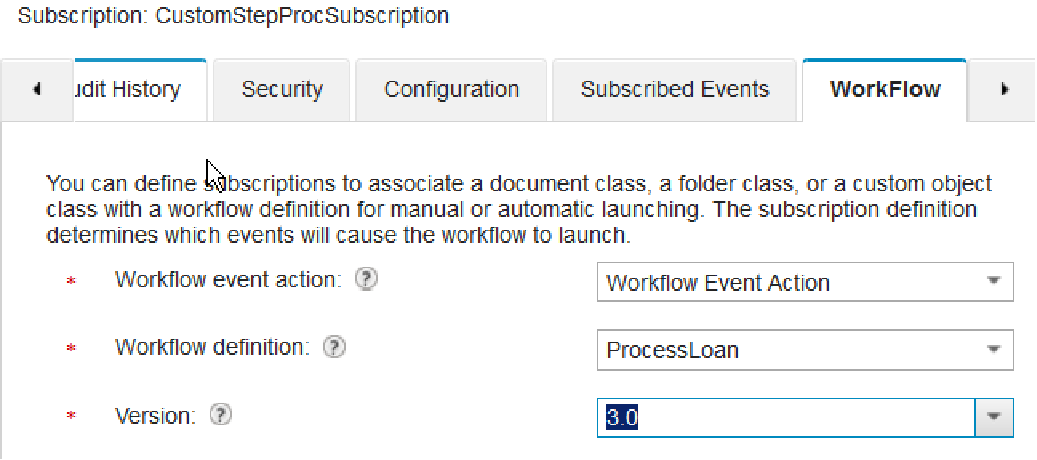
   a. Click the Workflow subtab. You might have scroll along the subtabs to see Workflow.



Subscription: CustomStepProcSubscription

   b. Observe the value for the Workflow definition field: `ProcessLoan`

   c. For the Version field, select the latest version from the list (Example: 3.0).



Subscription: CustomStepProcSubscription

You can define subscriptions to associate a document class, a folder class, or a custom object class with a workflow definition for manual or automatic launching. The subscription definition determines which events will cause the workflow to launch.

* Workflow event action:	Workflow Event Action
* Workflow definition:	ProcessLoan
* Version:	3.0

3. In the Workflow subtab, scroll down to the Property Maps section.

   a. Observe that the property names of the initiating document and data field names of the workflow are mapped.

   When you add a document, values for the workflow data fields are set as mapped in this step.

Note

The property names of the initiating document and data field names of the workflow can be different but their data types must match. For this lab, same names are used for convenience.

   b. Click Save and Close.

4. Log out of the Administration Console for Content Platform Engine application and close the browser.

Procedure 3: Test the custom step processor

In this procedure, you test the custom step processor for the customization that you did so far. When you add a document of LoanApp class, it launches the workflow that is associated with it. Since you configured the custom step processor in this workflow, when you open the VerifyInfo step, the step opens in the custom step processor that has an embedded viewer.

1. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In the Browse view, add a document with the sample data that is provided in the following table.

Item or Property name	Value
Repository	LoanProcess
Folder	Loan Applications
Document class	Loan > LoanApp
Local document	C:\ICN\SampleDocs\CarolCookLoan.pdf
Document Title	Carol Cook Loan App
CustomerName	Carol Cook
LoanAmount	250000

The workflow that is associated with this document class is launched automatically.

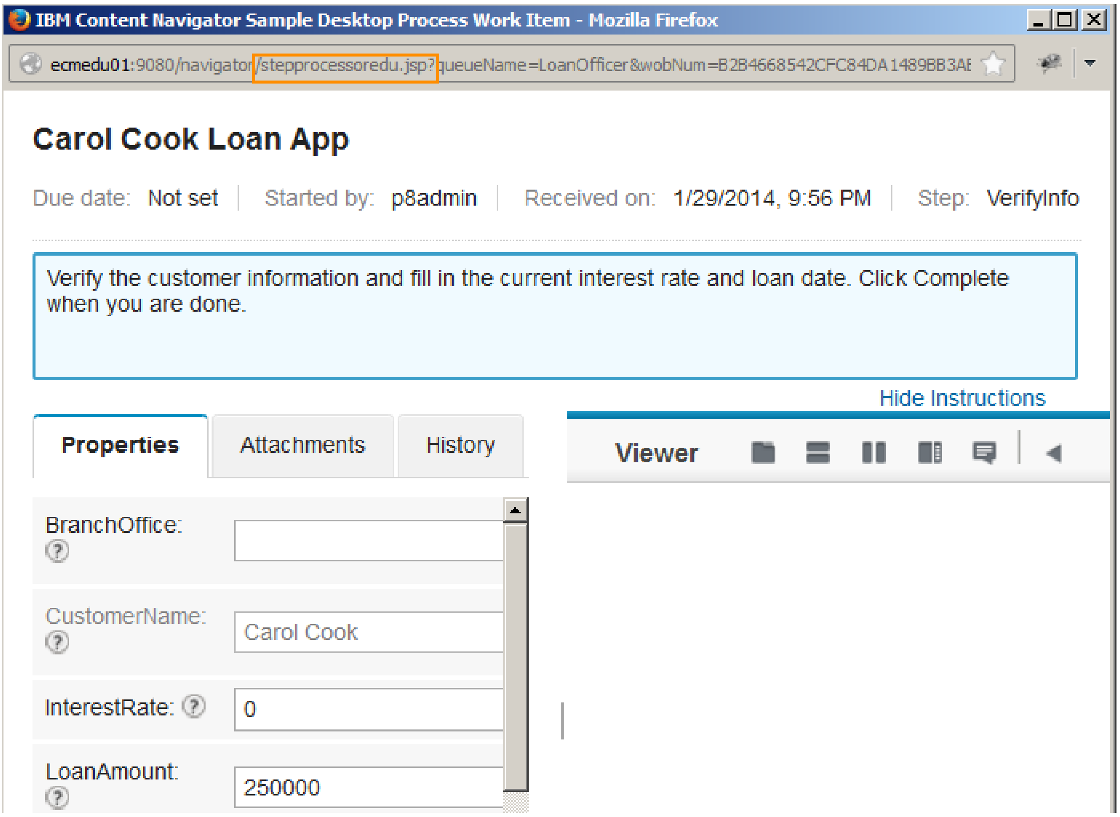
3. Click the double-arrrow icon to go to the Work view.
   a. In the Workflow view, select the Loan Process repository.
4. Expand Loan Officer role and click LoanVerification in-basket.
5. Double-click the work item with the Customer Name “Carol Cook” to open it.

Note

If you do not see your work item, you must refresh the browser.

6. Verify that the step is opened with the custom step processor that you created.
7. The URL contains the step processor that you created.

8. The user interface has the embedded Viewer.



IBM Content Navigator Sample Desktop Process Work Item - Mozilla Firefox
ecmedu01:9080/navigator/stepprocessoredu.jsp?queueName=LoanOfficer&wobNum=B2B4668542CFC84DA1489BB3AE
Properties
Attachments
History
Viewer
Hide Instructions

Carol Cook Loan App

Due date: Not set | Started by: p8admin | Received on: 1/29/2014, 9:56 PM | Step: VerifyInfo

Verify the customer information and fill in the current interest rate and loan date. Click Complete when you are done.

BranchOffice:

CustomerName: Carol Cook

InterestRate: 0

LoanAmount: 250000

b. Close the step processor by clicking Cancel.

c. Log out of IBM Content Navigator and close the browser.

Note

In the next lesson, you add an action to open the attached document in the embedded viewer.

Exercise 3.1.6: Add External Data Services to step processor

Introduction

External data services can be configured for the default workflow step processors and custom step processors. This configuration enables the users to add data validation, choice lists and manage other property behaviors for the step processor data fields.

You can reuse the external data services that you implemented in the “Implement External Data Services” unit. The files need to be updated to include the workflow step processor as a target.

Procedures

Procedure 1: Update ObjectTypes.json

Procedure 2: Add the JSON file

Procedure 3: Prepare the SampleEDSService application

Procedure 4: Test the external data service in the custom step processor

Procedure 1: Update ObjectTypes.json

In this procedure, you update the ObjectTypes.json file to include your workflow that has the custom step processor.

1. Open Eclipse by double-clicking the Eclipse_Kepler icon in your desktop.
   a. In the Workspace Launcher window, leave the default workspace directory (`C:\ICN\workspace_Kepler`) and click OK.
   b. In Eclipse, expand the sampleEDSService project in the Package Explorer on the left pane.
2. Open the ObjectTypes.json file.
   a. Expand Java Resources > src.
   b. Double-click the `ObjectTypes.json` file to open.
3. Edit the file.
   a. Add the following line to the top of the file after the open square bracket symbol`[`.

```
{"symbolicName": "ProcessLoan.Workflow.VerifyInfo"},
```

Note

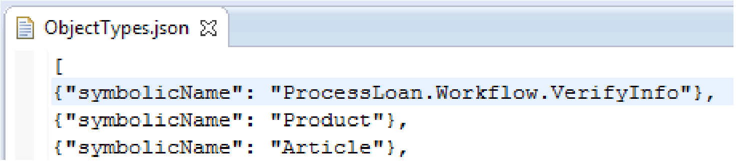
`ProcessLoan` is the workflow name. If there are any spaces in the workflow name, you must replace the spaces with underscores. `VerifyInfo` is the step name in the workflow.

   b. Make sure to add a comma at the end of the line.

Add External Data Services to step processor

(Continued)

c. The text looks like the following screen capture.



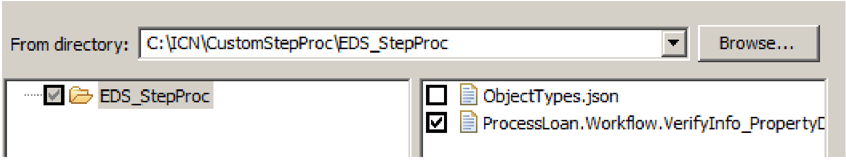
```
[
{"symbolicName": "ProcessLoan.Workflow.VerifyInfo"},
{"symbolicName": "Product"},
{"symbolicName": "Article"},
```

4. Save and close the file.

Procedure 2: Add the JSON file

In this procedure, you add a JSON file to represent the data fields in your workflow.

1. Add the solution file.
   a. In the Package Explorer on the left pane, expand Java Resources > src.
   b. Right-click the src node, and select Import from the list.
   c. In the Import window, select General > File System and click Next.
   d. Click Browse.
   e. In the "Import from directory", select the `C:\ICN\CustomStepProc\EDS_StepProc` folder and click OK.
   f. In the Import window, select the `ProcessLoan.Workflow.VerifyInfo_PropertyData.json` file.



From directory: C:\ICN\CustomStepProc\EDS_StepProc

   g. Click Finish.
2. Open the `ProcessLoan.Workflow.VerifyInfo_PropertyData.json` file.
   a. In the Package Explorer, double-click the file that you just added.
3. Check the file for the following property entries:
   – LoanNumber - Validate the format of this field.
   – Region - Add a choice list.
   – BranchOffice - Makes this choice list a dependent property of Region.
4. Close the file without any changes.

Add External Data Services to step processor

(Continued)

Procedure 3: Prepare the SampleEDSService application

Do the following tasks.

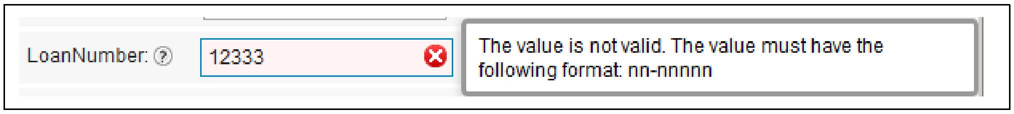
Note

Refer to the “Exercise 3.1.3: Set the input field status as required” in the “Unit.3. Implement External Data Services”, if you need detailed walk-through instructions for each of these tasks.

1. Build a WAR file for the SampleEDSService application.
2. Deploy the SampleEDSService application in WebSphere Application Server.
3. Verify the SampleEDSService application deployment.
4. Reload the EDS as plug-in.

Procedure 4: Test the external data service in the custom step processor

1. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In the Workflow view, select the Loan Process repository.
3. Expand Loan Officer role and click LoanVerification in-basket.
4. Double-click the work item with the customer name Carol Cook to open it.
5. In the step processor, select the Properties tab.
6. Verify that when you enter a value for the LoanNumber field, if the value is not in the “nn-nnnnn” (Example: 12-12345) format, an error is shown.



LoanNumber: 12333

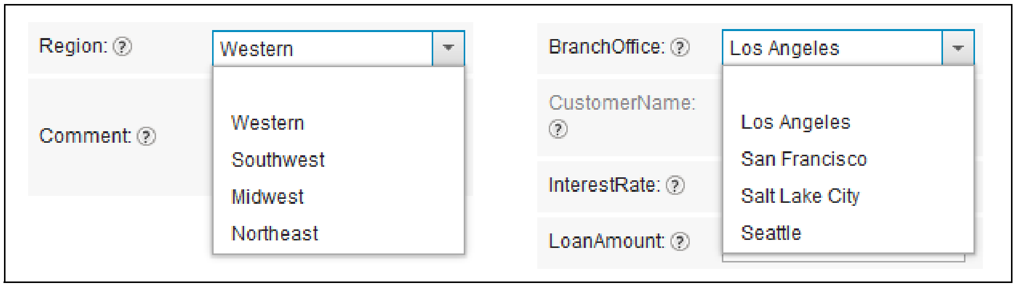
The value is not valid. The value must have the following format: nn-nnnnn

   a. Enter a value with the correct format (Example: 12-12345). Notice that the error is not shown.

Add External Data Services to step processor

(Continued)

7. Verify that the Region field has a choice list. Since it is a dependent choice list, when you select a value for the Region, the choice list values for the BranchOffice are updated.



Region: Western

BranchOffice: Los Angeles

b. Close the step processor by clicking Cancel.

c. Log out of IBM Content Navigator and close the browser.

LESSON 3.2: Add an Action to the Custom Step Processor

What this lesson is about?

This lesson describes how to add an action to the custom step processor action menu.

What you should be able to do?

After completing this lesson, you should be able to:

- Add an action to the custom step processor action menu.

How you will check your progress?

- Hands on Labs

Lab Solution files

- The solution files for this lesson are included in the following folder:
  C:\ICN\StepProc

Add an action to use embedded viewer

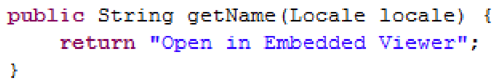
Add action plug-in overview

- To add an action, a Content Navigator plug-in is created in your student image.
- Following files are used for this plug-in.
  - StepProcessorActionPlugin.java
  - StepProcessorAction.java
  - StepProcessorActionPlugin.js

StepProcessorAction.java

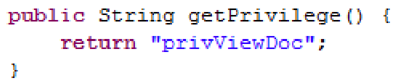
This file provides general information about the action that is implemented.

- The getName() method specifies the string that is shown on menus for the new action.



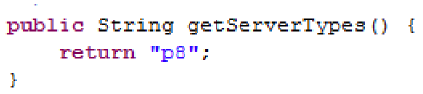
```
public String getName(Locale locale) {
    return "Open in Embedded Viewer";
}
```

- The getPrivilege() method specifies which privilege is required for this action. Anyone, who can view the document, can run this action.



```
public String getPrivilege() {
    return "privViewDoc";
}
```

- The getServerTypes() method must return the types of servers that are supported. This example works with IBM FileNet P8, so return "p8" as the value.

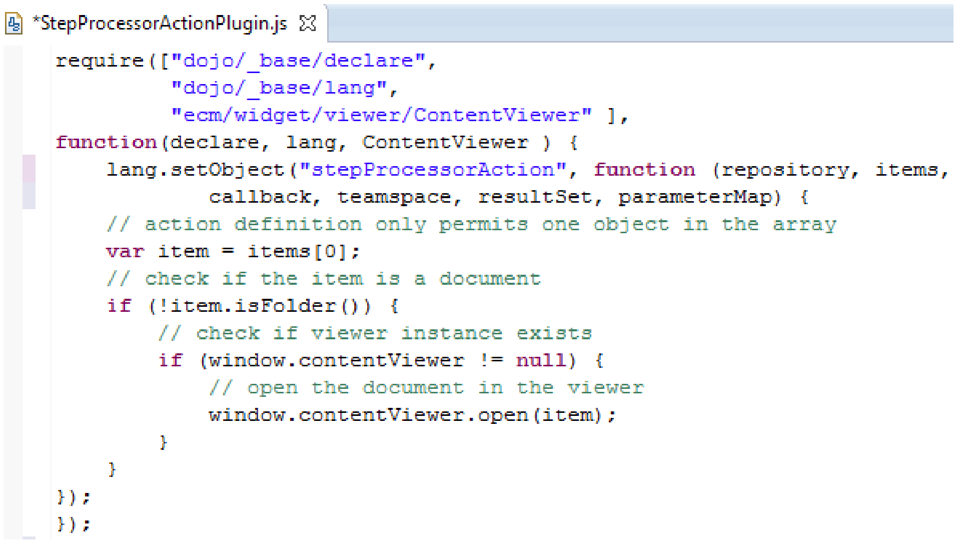


```
public String getServerTypes() {
    return "p8";
}
```

StepProcessorActionPlugin.js

This file contains the code for the action that is added.

- The code verifies if a document is selected.
- It then verifies that the embedded viewer is created.
- If both of these conditions are met, then it sends the document to the embedded viewer.



*StepProcessorActionPlugin.js

```
require(["dojo/_base/declare",
         "dojo/_base/lang",
         "ecm/widget/viewer/ContentViewer" ],
function(declare, lang, ContentViewer ) {
    lang.setObject("stepProcessorAction", function (repository, items,
            callback, teamspace, resultSet, parameterMap) {
    // action definition only permits one object in the array
    var item = items[0];
    // check if the item is a document
    if (!item.isFolder()) {
        // check if viewer instance exists
        if (window.contentViewer != null) {
            // open the document in the viewer
            window.contentViewer.open(item);
        }
    }
});
});
```

Exercise 3.2.1: Add an action to use the embedded viewer

Introduction

In the previous lab, you created the step processor with an embedded viewer. The Content Navigator step processor automatically shows the document in the external viewer.You must provide a method for users to see documents in the embedded viewer. You add an action that allows the document to be shown in the embedded viewer. The default Content Navigator behavior is not overridden so the users have the option of using the embedded or non-embedded viewer.

Procedures

Procedure 1: Check the source files for the plug-in

Procedure 2: Register the plug-in

Procedure 3: Add the action to a menu

Procedure 4: Assign the new menu to your desktop

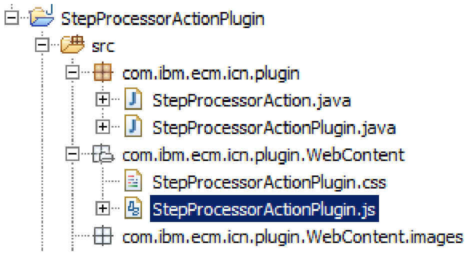
Procedure 5: Test the action in the custom step processor

Procedure 1: Check the source files for the plug-in

Note

The procedure to create a Content Navigator plug-in for an action is similar to the one you used in the "Develop Plug-ins" unit, "Develop Plug-in to Add an Action" lesson. A plug-in project for this exercise is already created and a JAR file is generated in Eclipse.

1. Optional step: View the source files for the plug-in.
   a. Open Eclipse_Kepler by double-clicking the shortcut.
   b. Check the code for the following files in the StepProcessorActionPlugin project.
      – StepProcessorAction.java
      – StepProcessorActionPlugin.js

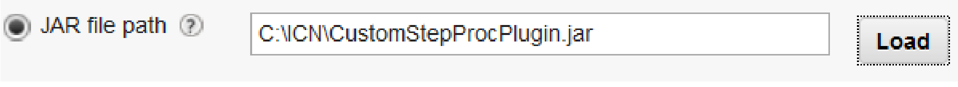


```
StepProcessorActionPlugin
  src
    com.ibm.ecm.icn.plugin
      StepProcessorAction.java
      StepProcessorActionPlugin.java
    com.ibm.ecm.icn.plugin.WebContent
      StepProcessorActionPlugin.css
      StepProcessorActionPlugin.js
    com.ibm.ecm.icn.plugin.WebContent.images
```

   c. Close the Eclipse.

Procedure 2: Register the plug-in

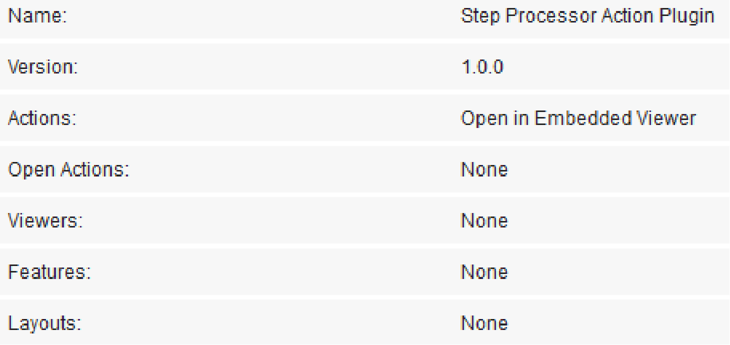
1. In your Firefox browser, open the IBM Content Navigator Admin Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=admin`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In the Admin desktop, click the Plug-ins icon in the left pane.
   a. In the Plug-ins tab, click the New Plug-in button.
   b. In the New Plug-in page, select the "JAR file path" option.
   c. Enter C:\ICN\CustomStepProcPlugin.jar for the field.



● JAR file path ? C:\ICN\CustomStepProcPlugin.jar

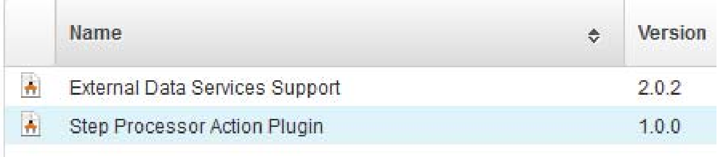
3. Click Load.

   If the file path is valid, the page shows more information as defined for the plug-in.



Name:	Step Processor Action Plugin
Version:	1.0.0
Actions:	Open in Embedded Viewer
Open Actions:	None
Viewers:	None
Features:	None
Layouts:	None

4. Click Save and Close.
5. Verify that the new plug-in is listed in the Plug-ins tab.



	Name	Version
	External Data Services Support	2.0.2
	Step Processor Action Plugin	1.0.0

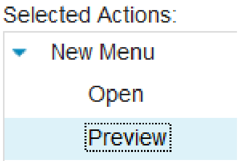
6. Close the Plug-ins tab.
7. Leave the admin desktop open for the next procedure.

Procedure 3: Add the action to a menu

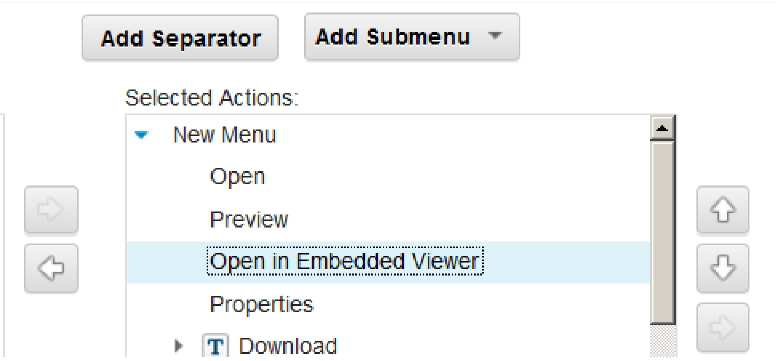
After the plug-in is registered, the action that is associated with the plug-in is available for adding to a menu.

1. The Admin desktop is already opened.
   a. Expand the Menus node in the left pane.
   b. Find the "Default attachment item context menu".
2. The default menu cannot be modified. Make a copy by right-clicking, and select Copy.

   This step creates a new menu of the same type.
3. Enter a name for the custom menu (Example: Custom Step Processor Viewer).
   a. The id is automatically populated based on the name you specify.
4. In the Selected Actions pane, select an item after which you want to show the custom action. (Example: Preview)



5. From the Available Actions on the left, select the "Open in Embedded Viewer" action and move it to the Selected Actions.



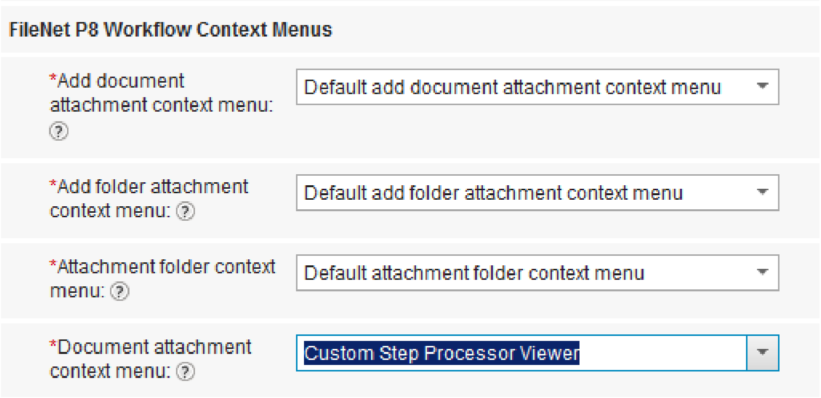
6. Click Save and close.
7. Leave the admin desktop opened for the next procedure.

Procedure 4:Assign the new menu to your desktop

In this procedure, you replace the existing default "Document attachment context menu" for this desktop with the new one.

1. In the Desktops tab, select Sample Desktop and click Edit.
2. Click the Menus tab and scroll down to the "FileNet P8 Workflow Context Menus" section.
3. Assign the new menu.

a. For the “Document attachment context menu”, select the new menu that you created (Custom Step Processor Viewer) from the list.



FileNet P8 Workflow Context Menus

*Add document attachment context menu: ?	Default add document attachment context menu
*Add folder attachment context menu: ?	Default add folder attachment context menu
*Attachment folder context menu: ?	Default attachment folder context menu
*Document attachment context menu: ?	Custom Step Processor Viewer

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

Procedure 5: Test the action in the custom step processor

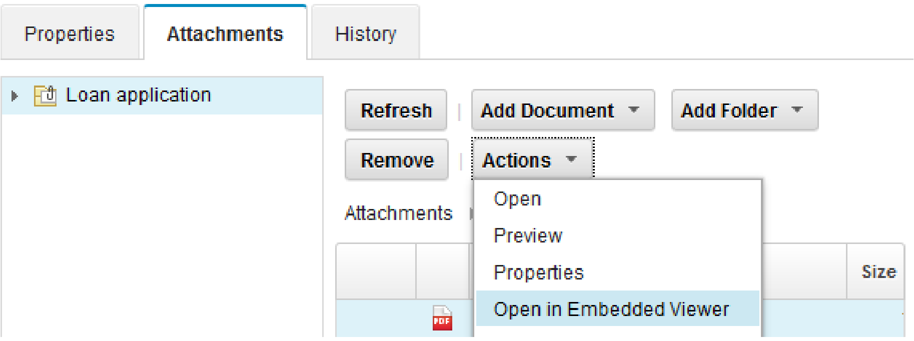
1. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
   - URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. In the Workflow view, select the Loan Process repository.
   a. Expand the Loan Officer role and click the LoanVerification process in-basket.
   b. Double-click the work item with the customer name Carol Cook to open it.
3. In the step processor, click the Attachments tab.
4. Select the Loan Application node. The document is listed in the Attachments > Loan application section.
5. Select the document.

Note

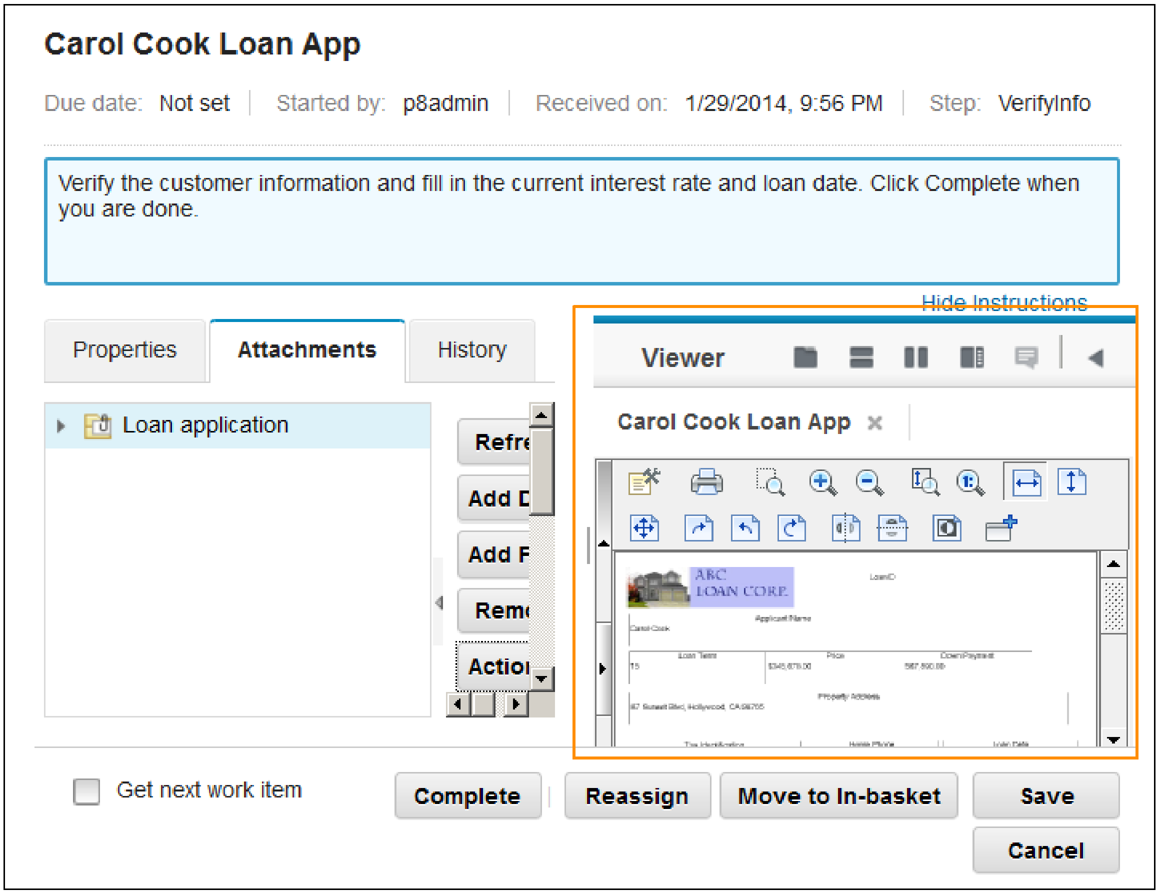
If you double-click the document, it opens in an external viewer as configured in the default step processor.

6. Click the down-arrow next to the Actions.

7. Verify that the “Open in Embedded Viewer” action is added to the menu.



8. Click the action and verify that the document opens in the embedded viewer.


Carol Cook Loan App

Due date: Not set | Started by: p8admin | Received on: 1/29/2014, 9:56 PM | Step: VerifyInfo

Verify the customer information and fill in the current interest rate and loan date. Click Complete when you are done.

9. Close the step processor by clicking Cancel.

10. Log out of IBM Content Navigator and close the browser.

4

Use Content Navigator Widgets in Other Applications

This unit describes how you can integrate IBM Content Navigator components into external applications.

LESSON 4.1: Use Content Navigator Widgets in Other Applications

What this lesson is about

This lesson describes how you can integrate IBM Content Navigator components into external applications.

What you should be able to do

After completing this lesson, you should be able to:

- Integrate IBM Content Navigator components into external applications with URL API.

How you will check your progress?

- Hands on labs.

References

IBM FileNet P8 5.2 Information Center

IBM DeveloperWorks forums

IBM Developer Works Technical Library

IBM Redbooks publication: Customizing and Extending IBM Content Navigator

Externalize IBM Content Navigator widgets

Integrate IBM Content Navigator into other applications

You must complete the following two steps to integrate Content Navigator or just some of the components into an external application.

1. Externalize IBM Content Navigator widgets so that it shows the parts to be integrated.
   - It can be either the entire IBM Content Navigator application or just some of its widgets.
   - It can be the standard layout of IBM Content Navigator or a custom layout with a different style.
   - This step can be independent of the final target system in which the IBM Content Navigator is integrated.
2. Integrate the externalized Content Navigator parts from step 1 into the external application.
   - It can be a simple URL invocation of IBM Content Navigator, for example, in an IFRAME of the external application.
   - It can be a tighter integration where the JavaScript code is running in the external application.

Options to externalize the IBM Content Navigator widgets

An external application can be a stand-alone application that starts components of IBM Content Navigator. The application can also be a container like a portal server that provides the runtime environment in which you run the IBM Content Navigator components.

- You can externalize the IBM Content Navigator widgets in the following ways:
  - Integrate Content Navigator with URL API.
  - Integrate Content Navigator with a specific feature.
  - Integrate Content Navigator with a specific layout.
  - Integrate stand-alone widgets in a Microsoft SharePoint page.
  - Integrate stand-alone widgets as a portlet in IBM WebSphere Portal.

Integration approaches

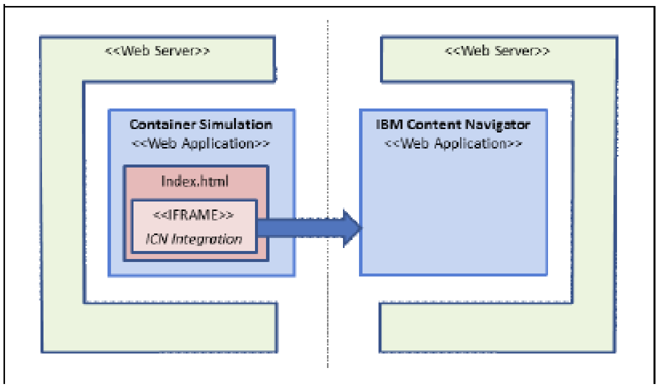
- The IBM Content Navigator externalization can be categorized into the following approaches.
  - **Unbound integration** - The external application starts a URL of the IBM Content Navigator application and the IBM Content Navigator widgets are running outside the external application.
  - **Bound integration** - This method uses widgets outside of the standard IBM Content Navigator application.

Unbound integration

Introduction

- The external application starts a URL of the IBM Content Navigator application and the IBM Content Navigator widgets are running outside the external application.

The diagram shows an unbound integration.



- The Container Simulation is a small web application that you can deploy in the same web server where you have the IBM Content Navigator deployment, or into a different web server.
- This simulation type is appropriate for the following approaches:
  - Integrate Content Navigator with URL API.
  - Integrate Content Navigator with a specific feature.
  - Integrate Content Navigator with a specific layout.

Externalize with URL API and deep linking

Introduction

- Use of IBM Content Navigator URL API is the easiest method. In this method, IBM Content Navigator is integrated as a whole application.
- From the external application, you start the URL of IBM Content Navigator.
- You can control what parts of IBM Content Navigator that are visible to the users.
  - Use more URL parameters with its URL API.
- You can render a particular feature or a desktop without the context of the entire desktop for integration through the iframe.

Advantages

- No coding is required. You must set up only the configuration.
- No cross-site scripting issues. You use iframe integration.
- No Dojo version conflicts between the external application and IBM Content Navigator.

Limitations

- You have the following limitations with iframe integration.
  - Limited interaction with the master application with the change of the URL.
  - Each invocation always reloads the whole desktop.
  - Complete IBM Content Navigator integration (not a specific part or several widgets).

Available URL types in IBM Content Navigator

- IBM Content Navigator allows you access desktops, features, documents, search templates and folders directly by adding a constructed URL link into your application.
- Following URL types are available in IBM Content Navigator:
  - Desktop URL - Opens a specified desktop in the full IBM Content Navigator client.
  - Feature URL - Opens a desktop with a specified feature preselected and ignores the default feature that is set for the desktop.
  - Folder URL - Opens a view that displays the subtree and contents of the specified folder.
  - Document URL - Opens the document in the preconfigured viewer for the MIME type of the document.

    If the document is a StoredSearch, it opens the search form.

Externalize with a custom Feature

Introduction

- To open a specific feature in IBM Content Navigator, you can specify the feature id in the URL.
- The standard feature ids are:
  - `browsePane` (for the standard browse feature)
  - `searchPane` (for the standard search feature)
  - `favorites` (for the standard favorite feature)
  - `manageTeamspaces` (for the standard teamspace feature)
  - `workPane` (for the standard work feature)

Custom Features

- If you want to externalize your own feature, you can provide a custom feature through a plug-in and start directly this feature with the URL API.
- If you want to show the feature without toolbars at the top and on the left side, you can specify the sideChrome parameter to the URL.
- Custom feature gives you complete freedom about which widgets to show and how to arrange them on the page.
  - All the IBM Content Navigator widgets and models can be repurposed, or you can develop your own widgets.

Advantages

- You can specify which widgets to render in a custom feature.
- Seamless integration with the sideChrome parameter.
- No cross-site scripting issues. You use iframe integration.
- No Dojo version conflicts between the external application and IBM Content Navigator.

Limitations

You have the following limitations when you use the specific feature approach.

- You can show or hide (but cannot change) the top banner or the left feature bar.
- You cannot change the default layout of the IBM Content Navigator.
- You cannot externalize only a specific widget.
- Limitations with the iframe integration apply to this approach also.

Externalize with a custom Layout

Introduction

In this approach, you can do the following tasks:

- Start the complete IBM Content Navigator with URL invocation of the standard launch page; like the first two approaches that uses URL API or custom feature.
- Change the standard layout of IBM Content Navigator (which is the NavigatorMainLayout widget) with a custom layout provided with a plug-in.
- Arrange the features that are based on your requirements or omit some features.

Advantages

- You can specify which widgets to render in a custom layout.
- You can customize the top banner or the left feature bar.
- Layout can contain several features and act as a container for all your integration features.
- No cross-site scripting issues. You use iframe integration.
- No Dojo version conflicts between the external application and IBM Content Navigator.

IBM Information Center documentation

More details about the construction of URL and available parameters are available:

IBM FileNet P8:

IBM Content Manager

IBM Content Manager OnDemand

Note

When you integrate the IBM Content Navigator application through URL, you can choose Single-Sign-On solution between your application and IBM Content Navigator. The user does not need to login to IBM Content Navigator when redirected from your application or when part of IBM Content Navigator is rendered within your application.

Bound integration

Introduction

- This method uses widgets outside of the standard IBM Content Navigator application.
  - You do not need to start the entire IBM Content Navigator with its standard initialization process.
  - URL invocation is not required.
- The external application initializes, starts, and interacts with the IBM Content Navigator widgets directly.
  - IBM Content Navigator widgets run inside the external application or container.

Required conditions for bound integration

- The target system must support the Dojo framework.
- The exact Dojo version that IBM Content Navigator uses, must work inside the external application.

Advantages

- Allows interaction with the master application and wiring with its components.
- Loads the desktop model of IBM Content Navigator only one time.
- Control the initialization process of the widgets to your needs.

Limitations

- Requires deep understanding of the IBM Content Navigator and its wiring.
- Requires more coding - initialization of the JavaScript Model.
- Cross-site scripting issues.
- Dojo version conflicts if the external application also uses the Dojo framework.

Demonstrations

Set up an external web application

Click here to watch the demonstration.

Exercise 4.1.1: Set up an external web application

Introduction

One of the ways to do an unbound integration of IBM Content Navigator into another application is to start its URL. You can invoke the start page of IBM Content Navigator without more parameters and the IBM Content Navigator shows as in the default application.

In this lab, you create and deploy an external web application to simulate the IBM Content Navigator integration.

Procedures

Procedure 1: Create a plug-in project in Eclipse

Procedure 2: Add a welcome file and a style sheet

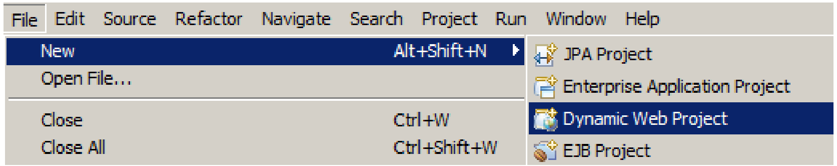
Procedure 3: Create a Web Archive (WAR) File

Procedure 4: Deploy the EDUContainerSimulation WAR File

Procedure 5: Test the external application

Procedure 1: Create a plug-in project in Eclipse

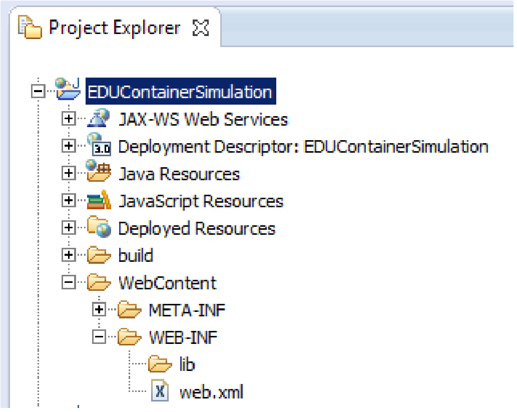
1. If it is not already started, start the WebSphere Application Server.
   – Double-click the Start Server.bat file in the WebSphere Admin folder on the desktop.
   – Wait for the Start the server page to close.
2. Open Eclipse by double-clicking the Eclipse_Kepler icon in your desktop.
   a. In the Workspace Launcher page, leave the default workspace directory (`C:\ICN\workspace_Kepler`) and click OK.
3. Open the project creation wizard.
   a. In Eclipse, click File > New > Dynamic Web Project.



   The New Dynamic Web Project page opens.
4. Create a Project.
   a. In the `New Dynamic Web Project` page, enter a Project Name (Example: `EDUContainerSimulation`) and click Next.
   b. Leave other default settings and click Next.

c. In the Web Module page, select “Generate web.xml deployment descriptor” and click Finish.

5. The wizard generates a web project with a web.xml.
6. Your project directories must look like the one that are shown in the screen capture.



Procedure 2: Add a welcome file and a style sheet

In this procedure, you add a welcome file and a style sheet to the project that you created.

1. Right-click the `EDUContainerSimulation > WebContent` folder and click Import.
   a. In the Import page, select General > File System and click Next.
   b. In the Import > File system page, click Browse.
   c. In the “Import from directory” page, go to the `C:\ICN\OtherApps\Ex.6.1.1` and click OK.
   d. Back in the Import > File system page, select `index.html` and `EDUContainerSimulation.css`.
2. Make sure that the “`Into folder`” field has the following value: `EDUContainerSimulation > WebContent`
   a. Select the “Overwrite existing resources without warning” option.
   b. Click Finish.
3. Open the `index.html` file and observe the contents:
   - The index.html welcome page is structured with three DIV tags. The first and the last one represent content of the external application.
   - The middle DIV tag contains content of the externalized IBM Content Navigator.
     – You do the integration with an <`iframe`> element (unbound integration).

- The URL in the `src` attribute points to your IBM Content Navigator deployment.
  – You can adapt the URL in the src attribute for the different approaches:
  – For a simulation of the bound integration, the externalized IBM Content Navigator widget is initialized directly in the middle DIV tag that simulates a direct integration into the external application.

4. Close the file without changes.

Procedure 3: Create a Web Archive (WAR) File

In this procedure, you pack the web application to a WAR file with the Eclipse tool.

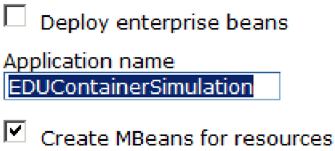
1. Right-click the `EDUContainerSimulation project` and click Export > WAR file from the list.
   a. In the Export page, the Web project field has the value pre-filled automatically.
   b. For the Destination field, click Browse.
   c. In the “Save as” page, go to C:\ICN folder.

      The file name (`EDUContainerSimulation.war`) is pre-filled automatically.
   d. Click Save.
   e. Back in the Export > War Export page, optionally, select “Export source files” and “Overwrite existing file” and click Finish.
   f. Leave Eclipse open through out this unit.

Procedure 4: Deploy the EDUContainerSimulation WAR File

In this procedure, you deploy the WAR file to WebSphere Application Server where IBM Content Navigator is deployed.

1. In a Firefox browser, go to the WebSphere Application Server admin tool.
   a. Click the WAS Admin link in the Bookmarks or go to the following URL: `https://ecmedu01:9043/ibm/console/logon.jsp`
   b. Log in as an Administrator.
      – User ID: p8admin
      – Password: IBMFileNetP8
2. Deploy the application.
   a. In the left pane, expand Applications and click New Application.
   b. Click “New Enterprise Application” in the New Application page in the right pane.
3. In the next page, accept the default option (Local file system) and click Browse.
   a. In the File Upload page, go to the C:\ICN folder and select the War file that you created (`EDUContainerSimulation.war`).
   b. Click Open.
   c. Back in the “Preparing for the application installation” page, click Next.
4. In the next page, accept the default option (Fast Path) and click Next.
5. In the Install New Application page > Select installation options, edit the value for the “Application name” field to `EDUContainerSimulation` and click Next.

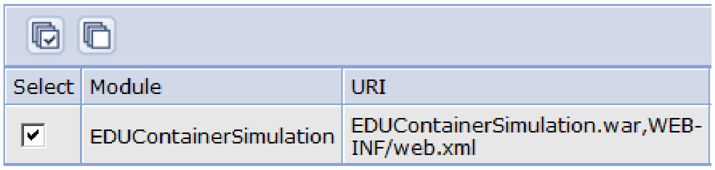


□ Deploy enterprise beans

Application name
EDUContainerSimulation

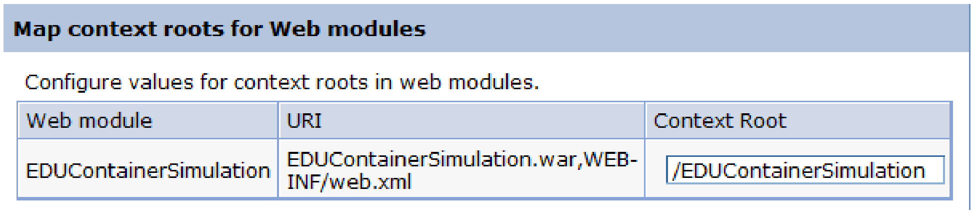
☑ Create MBeans for resources

6. In the “Map modules to servers” page, select the check box next to your module and click Next.



Select	Module	URI
☑	EDUContainerSimulation	EDUContainerSimulation.war,WEB-INF/web.xml

7. In the “Map virtual hosts for Web modules” page, select the check box next to your module and click Next.
8. In the “Map context roots for Web modules” page, enter /`EDUContainerSimulation` for the Context Root and click Next.



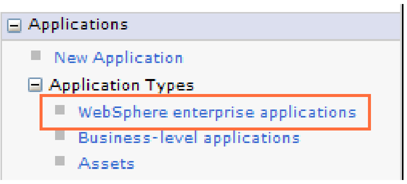
Map context roots for Web modules

Configure values for context roots in web modules.

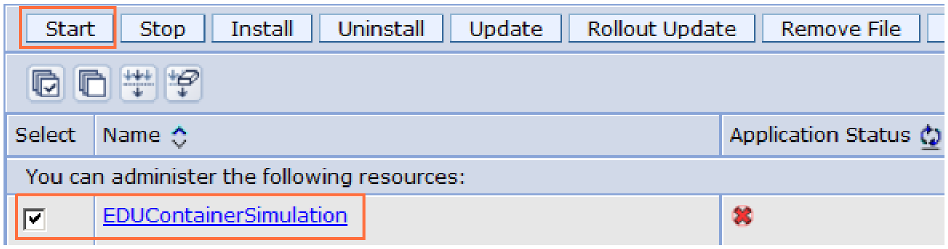
Web module	URI	Context Root
EDUContainerSimulation	EDUContainerSimulation.war,WEB-INF/web.xml	/EDUContainerSimulation

9. In the “Metadata for modules” page, select the check box for the metadata column and click Next.
10. In the Summary page, click Finish.

    It takes a few moments to complete the installation.
11. Save the application.
    a. In the results page, scroll down and click the Save link, to save the changes to the master configuration.
12. Start the application.
    a. In the left pane, expand Applications > Applications Types.
    b. Click the WebSphere enterprise applications link.


Applications
New Application
Application Types
WebSphere enterprise applications
Business-level applications
Assets

    c. Select EDUContainerSimulation in the Enterprise Applications page in the right pane and click Start.



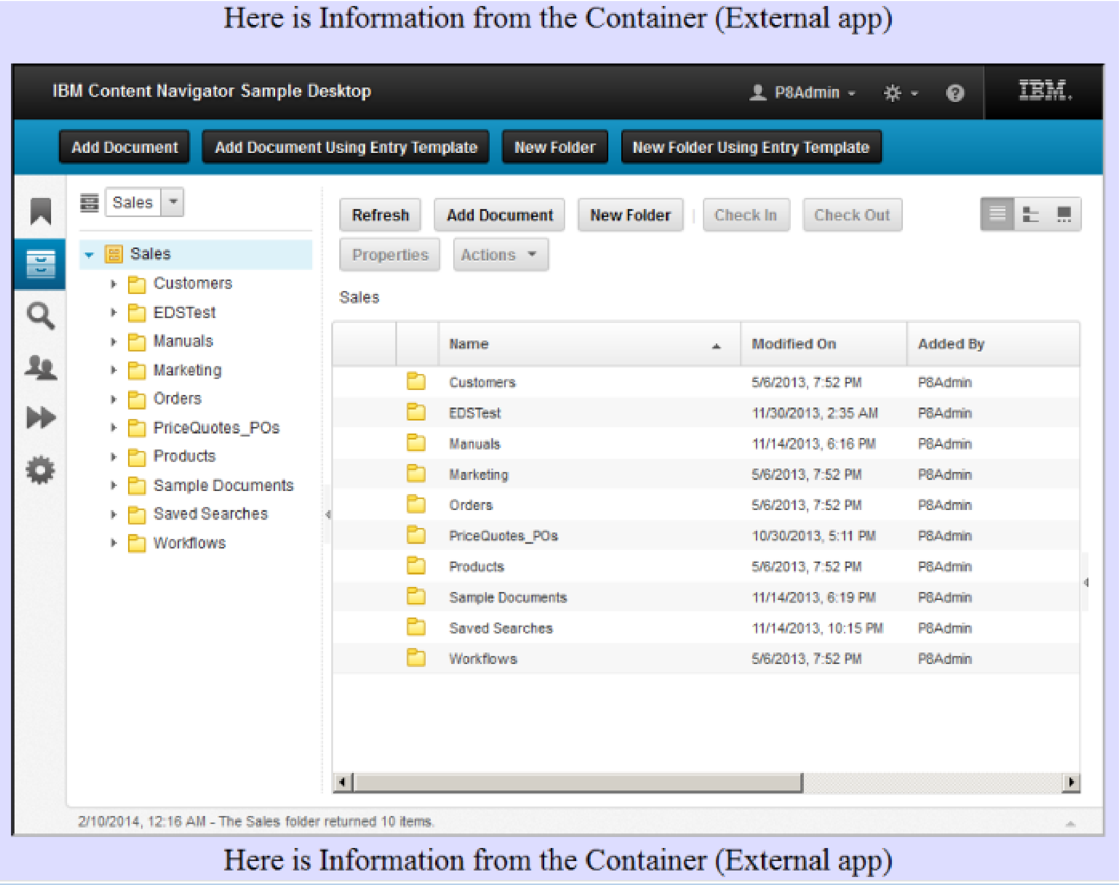
Select	Name	Application Status
You can administer the following resources:		
☑	EDUContainerSimulation	✖

13. Wait for the Start message to show at the top of the page.
14. Log out of the WebSphere Application Server admin tool and close the browser.

Procedure 5: Test the external application

In this procedure, you access the web application that you deployed and test it.

1. In a Firefox browser, start the external web application.
   - URL: `http://ecmedu01:9080/EDUContainerSimulation`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Verify that the new application shows the external content that you have in the index.html file and IBM Content Navigator.



Here is Information from the Container (External app)

Sales

Name	Modified On	Added By
Customers	5/6/2013, 7:52 PM	P8Admin
EDSTest	11/30/2013, 2:35 AM	P8Admin
Manuals	11/14/2013, 6:16 PM	P8Admin
Marketing	5/6/2013, 7:52 PM	P8Admin
Orders	5/6/2013, 7:52 PM	P8Admin
PriceQuotes_POs	10/30/2013, 5:11 PM	P8Admin
Products	5/6/2013, 7:52 PM	P8Admin
Sample Documents	11/14/2013, 6:19 PM	P8Admin
Saved Searches	11/14/2013, 10:15 PM	P8Admin
Workflows	5/6/2013, 7:52 PM	P8Admin

2/10/2014, 12:16 AM - The Sales folder returned 10 items.

Here is Information from the Container (External app)

3. Explore the IBM Content Navigator features.
4. Log out of the application and close the browser.

Exercise 4.1.2: Integrate Content Navigator with URL API

Introduction

IBM Content Navigator allows URL API that provides some control of what to show to the user through deep linking. You can specify the elements such as desktop, feature, folder, and document. In the previous exercise, you showed the content for the entire desktop. In this lab exercise, you show only the content of a specific folder of a repository that is associated with IBM Content Navigator.

Procedures

Procedure 1: Construct a URL for a folder

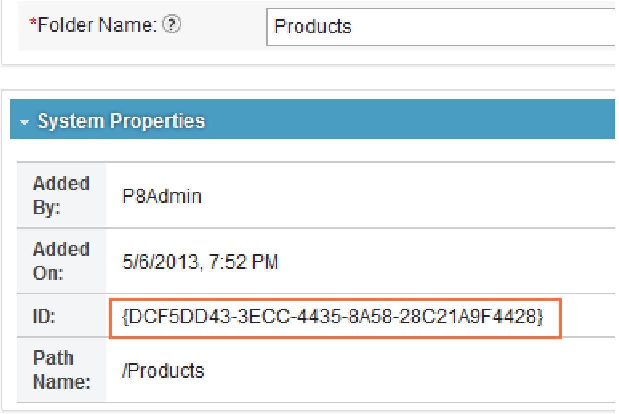
Procedure 2: Add the URL to the Simulation Container

Procedure 4: Test the external application

Procedure 4: Test the external application

Procedure 1: Construct a URL for a folder

1. Get the GUID value for a folder.
   a. In a Firefox browser, open the IBM Content Navigator Sample Desktop.
      – URL: `http://ecmedu01:9080/navigator/?desktop=SampleDesktop`
      – User name: `P8admin`
      – Password: `IBMFileNetP8`
   b. In the Browse view, select the Sales object store > Products folder in the left pane.
   c. Right-click the folder and select Properties from the list.
   d. In the Properties page, expand the System Properties section.



*Folder Name: Products

System Properties

Added By:	P8Admin
Added On:	5/6/2013, 7:52 PM
ID:	{DCF5DD43-3ECC-4435-8A58-28C21A9F4428}
Path Name:	/Products

   e. If the System Properties section is not visible, expand the size of the window.

   f. Copy the ID (GUID) value and paste it in a text editor (Notepad).

2. Copy the base URL for the navigator from the browser.
3. Construct a URL with the GUID value as shown in the following text.
   a. Make sure that you remove the curly braces in the URL.

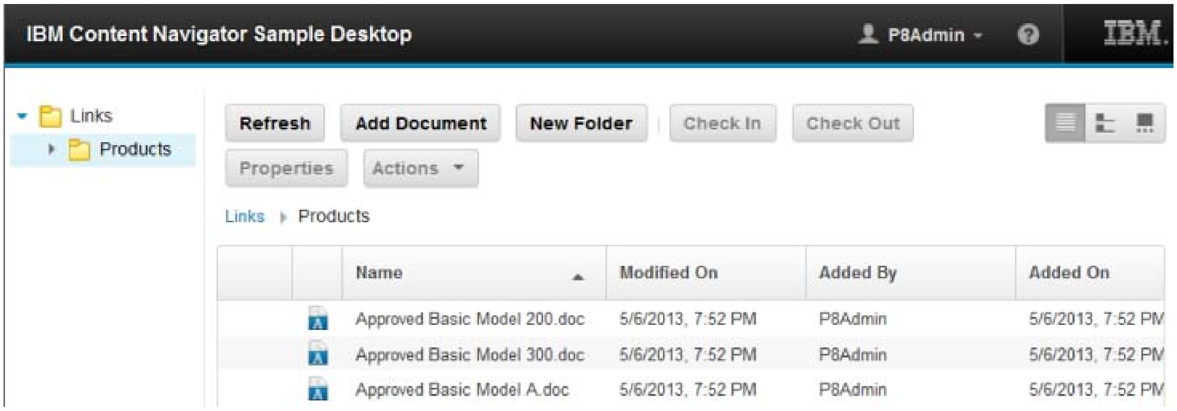
```
http://ecmedu01:9080/navigator/bookmark.jsp?desktop=SampleDesktop&feature=
browsePane&docid=DCF5DD43-3ECC-4435-8A58-28C21A9F4428
```

Note

Optionally, copy the text from the C:\ICN\OtherApps\Ex.6.1.2\FolderURL.txt file.

The docid parameter in the URL specifies the unique identifier of the folder. For the IBM FileNet P8 repositories, it is the property ID that is a Globally Unique Identifier (GUID) of the object.

4. Verify the constructed URL.
   a. Copy and paste the URL that you constructed into a browser.
   b. Log in when prompted:
      – User name: P8admin
      – Password: IBMFileNetP8
   c. You get a view that is similar to the one in the screen capture.
   d. Click the Links > Products in the left pane to see the folder contents in the right pane.
   e. Verify that the left pane contains no other features.

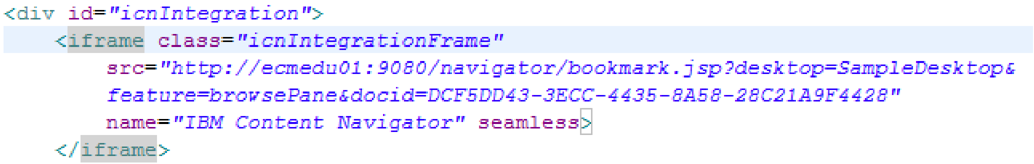


Name	Modified On	Added By	Added On
Approved Basic Model 200.doc	5/6/2013, 7:52 PM	P8Admin	5/6/2013, 7:52 PM
Approved Basic Model 300.doc	5/6/2013, 7:52 PM	P8Admin	5/6/2013, 7:52 PM
Approved Basic Model A.doc	5/6/2013, 7:52 PM	P8Admin	5/6/2013, 7:52 PM

Procedure 2: Add the URL to the Simulation Container

In this procedure, you add the URL that you constructed to the Simulation Container application from the previous exercise.

1. If it is not already open, open Eclipse by double-clicking the Eclipse_Kepler icon in your desktop.
   a. In the Workspace Launcher page, leave the default workspace directory (`C:\ICN\workspace_Kepler`) and click OK.
2. Edit the index.html file.
3. Replace the value for the `src` attribute of the `iframe` tag in `index.html` with the URL that you constructed for the folder.



```
<div id="icnIntegration">
    <iframe class="icnIntegrationFrame"
        src="http://ecmedu01:9080/navigator/bookmark.jsp?desktop=SampleDesktop&
        feature=browsePane&docid=DCF5DD43-3ECC-4435-8A58-28C21A9F4428"
        name="IBM Content Navigator" seamless>
    </iframe>
```

4. Save the file.
5. Create a Web Archive (WAR) File with the procedure from the previous exercise.

Procedure 3: Deploy the EDUContainerSimulation WAR File

In this procedure, you delete the existing application in the WebSphere Application Server and deploy the new EDUContainerSimulation application that you created.

1. In a Firefox browser, log in to the WebSphere Application Server Administration tool.
   a. Click the WAS Admin link in the Bookmarks or go to the following URL: `https://ecmedu01:9043/ibm/console/logon.jsp`
   b. Log in to the WebSphere Application Server Administration tool.
      – User ID: p8admin
      – Password: IBMFileNetP8
2. Stop the application.
   a. In the left pane, expand Applications > Applications Types.
   b. Click the WebSphere enterprise applications link.
   c. Select EDUContainerSimulation in the Enterprise Applications page in the right pane and click Stop.
   d. Wait for the stop message to display.
3. Delete the existing application.
   a. Select EDUContainerSimulation again and click “Uninstall” in the toolbar.
   b. In the “Uninstall Application” page, click OK.

c. In the Enterprise Applications page, click the Save link.

4. Deploy the new application.
5. Repeat the “Procedure 4: Deploy the EDUContainerSimulation WAR File” in the previous exercise to complete the deployment.

Procedure 4: Test the external application

1. Open the Container Simulation application in a Firefox browser.
   - URL: `http://ecmedu01:9080/EDUContainerSimulation`
   - User name: `P8admin`
   - Password: `IBMFileNetP8`
2. Click the Links > Products in the left pane to see the folder contents in the right pane.
3. Verify that the folder content is shown inside the iframe. The iframe is inside the simulated external application.
4. If the change is not reflected, clear the browser cache and test.
5. Log out and close the browser.

Exercise 4.1.3: Integrate Content Navigator with a Feature

Introduction

In this lab exercise, you integrate parts of IBM Content Navigator into an external application to show a single feature of IBM Content Navigator. You use IBM Content Navigator URL API to complete this task.

You also use an extra URL parameter `sideChrome` to hide the bar for features on the left side and the top banner for a more seamless integration.

Procedures

Procedure 1: Construct a URL for a feature

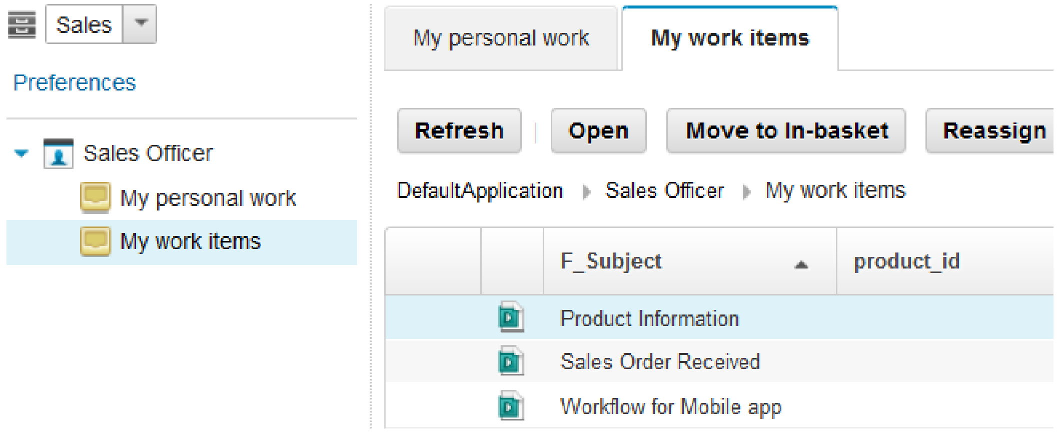
Procedure 2: Optional - Test the URL in the EDUContainerSimulation application

Procedure 1: Construct a URL for a feature

1. In Notepad, construct a URL with the feature id value as shown in the following text.

   ```
   http://ecmedu01:9080/navigator/?desktop=ExternalApp&feature=workPane&sideChrome=0
   ```

   Refer to the text in the C:\ICN\OtherApps\Ex.6.1.3\FeatureURL.txt file.
2. Verify the constructed URL.
   a. Copy and paste the URL that you constructed into a browser.
   b. Log in when prompted:
      – User name: `P8admin`
      – Password: `IBMFileNetP8`
   c. You get a view that is similar to the one in the screen capture.
   d. Click Sales Officer > My work items in the left pane to see the contents in the right pane.



DefaultApplication ▸ Sales Officer ▸ My work items

F_Subject	product_id
Product Information	
Sales Order Received	
Workflow for Mobile app	

e. Verify that the bar for features on the left pane and the top banner are hidden.

The sideChrome parameter hides the bar for features and the top banner.

Procedure 2: Optional - Test the URL in the EDUContainerSimulation application

1. Refer to the Procedure 2: Add the URL to the Simulation Container to Procedure 4 and do the following steps.
   - Add the URL (that you created for the feature) to the Simulation Container application.
     - Edit the index.html file in Eclipse.

       The solution index.html file is available in the C:\ICN\OtherApps\Ex.6.1.3 folder.
     - Replace the value for the `src` attribute of the `iframe` tag.
   - Create a Web Archive (WAR) File.
   - Deploy the new EDUContainerSimulation application in WebSphere Application Server.
     - Delete the existing application in WebSphere Application Server, and deploy.
   - Test the external application in a browser.

Additional Information

- In this exercise, you created a URL that shows the application spaces. When you open it, you can select an in-basket that shows you a list of work items.
  - In this example, the business user does two steps: open the application space and select the in-basket.
- If you want the users to directly open a specific inbox, you must extend the URL API of IBM Content Navigator.
- You can also develop your own work feature that can read more URL parameters such as ApplicationSpace and Inbasket, and directly open the work item list.
  - You must deploy the custom work feature as a plug-in that implements the feature extension point.

Note

Refer to the “Customizing and Extending IBM Content Navigator” RedBook for more details on this topic using the URL: http://www.redbooks.ibm.com/redpieces/abstracts/sg248055.html

5

Appendix

This Appendix contains supplemental material not included in the main course material.

APPENDIX A:Start and Stop System Components

Procedures

Procedure 1: Start WebSphere Application Server

Procedure 2: View the Content Engine Startup Context page

Procedure 3: Open the Content Platform Engine log files

Procedure 4: View the Process Engine Server information (Ping Page)

Procedure 5: View web applications

Procedure 6: Shutdown individual components

Procedure 7: Restart the system components

Procedure 1: Start WebSphere Application Server

WebSphere Application Server hosts the following applications:

- Tivoli Directory Server Administration tool
- Content Platform Engine
- IBM Content Navigator
- Administration Console for Content Platform Engine
- FileNet Workplace XT

1. Click Start > All Programs > IBM WebSphere > IBM WebSphere Application Server V8.5 > Profiles > AppSrv01 > Start the server.
2. Wait for the Start the server window to close.

Note

You can also use the Start the Server batch file in the WebSphere Admin folder on the desktop to start WebSphere Application Server.

Procedure 2: View the Content Engine Startup Context page

The Content Engine Startup Context (Ping Page) provides detailed information about the Content Platform Engine. You can use the Ping Page to quickly determine whether the Content Platform Engine is running.

1. Use Firefox to open the Content Engine Startup Context page.
   - http://ecmedu01:9080/FileNet/Engine
   - In general, the URL is http://<server>:<port>/FileNet/Engine.

2. Verify the following information:
   - Product name: P8 Content Platform Engine - 5.2.0
   - Log file location: C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSvr01\FileNet\server1
   - P8 Domain: P8Domain
   - JDBC Driver: IBM DB2 JDBC Universal Driver Architecture 3.62.56

Procedure 3: Open the Content Platform Engine log files

If startup problems occur, the Content Platform Engine log files provide information for troubleshooting.

1. Go to the location of the log files that you identified in the previous procedure and view them in WordPad or Notepad++.
   - p8_server_error.log
   - p8_server_trace.log
   - pesvr_system.log
   - pesvr_trace.log

Note

The Content Platform Engine has two main services: Content Services and Process Services. Both services create log files in the same location. Content Service log files begin with “p8,” while Process Service log files begin with “pesvr.”

Procedure 4: View the Process Engine Server information (Ping Page)

The Process Service also has a ping page.

1. Use Firefox to open the Process Engine Server information (Ping Page).
   - http://ecmedu01:9080/peengine/IOR/ping
   - In general, the URL is http://<server>:<port>/peengine/IOR/ping.
2. If prompted, log on as P8Admin.
   - User name: `P8Admin`
   - Password: `IBMFileNetP8`
3. Review the information on this page.
   - The Process Engine ping page includes Helpful links at the bottom of the page.
   - The database row shows database connection information.

Procedure 5: View web applications

You can view the applications that are running in the administrative console. From there, you can also shut down and restart any web application individually.

1. Use Firefox to open the administrative console:
   - https://ecmedu01:9043/ibm/console
   - In general, the URL is https://<server>:<port>/ibm/console
2. Log in to WebSphere Integrated Solutions Console as the `P8Admin` user.
   - Password: `IBMFileNetP8`
3. In the navigation pane, go to Applications > Application Types > WebSphere enterprise applications.
4. Verify that the following applications are running (shows a green arrow in the Application Status column):

Application name	**Purpose**
FileNetEngine	Content Platform Engine
IDSWebApp	Directory Services Web Administration console.
WorkplaceXT	FileNet Workplace XT
ivtApp	WebSphere installation verification test application
navigator	IBM Content Navigator
query	WebSphere query service
SampleEDSService	A sample external data service

Procedure 6: Shutdown individual components

In this procedure, you shut down the components individually. You are viewing the Enterprise Applications in the WebSphere Integrated Solutions Console.

1. Stop FileNet Workplace XT and IBM Content Navigator:
   a. In WebSphere Integrated Solutions Console, select the FileNet Workplace XT and IBM Content Navigator applications.
   b. Click Stop.
   c. Wait for the Application Status to show that it stopped.
2. Stop FileNetEngine.
3. Log out of WebSphere integrated Solutions Console.
4. Stop WebSphere Application Server:
   a. Click Start > All Programs > IBM WebSphere > IBM WebSphere Application Server V8.5 > Profiles > AppSrv01 > Stop the server.

b. Enter the following user name and password to stop the server:
   - User name: `P8Admin`
   - Password: `IBMFileNetP8`

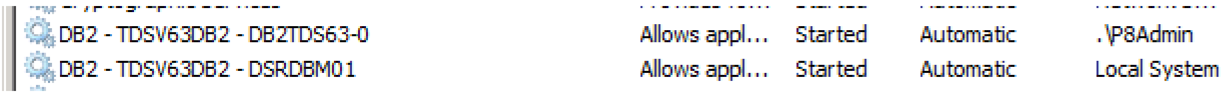
Note

You can also use Stop the Server.bat in the WebSphere Admin folder on the desktop. The batch file enters the user name and password for you.

5. Stop the database server software for the Content Platform Engine:

   a. Open the Windows Services Console: Click Start > Services.

   b. Locate the two DB2 server instances:



DB2 - TDSV63DB2 - DB2TDS63-0	Allows appl...	Started	Automatic	.\P8Admin
DB2 - TDSV63DB2 - DSRDBM01	Allows appl...	Started	Automatic	Local System

   c. Stop these two database services.

Note

IBM Tivoli Directory Server Instance V6.3 - dsrdbm01 stops when you stop the directory database instance.

Procedure 7: Restart the system components

1. Start the Directory Server and Content Platform Engine database DB2 server instances in this order:

   a. DB2 - TDSV63DB2 - DSRDBM01 (Directory Services DB2 instance)

   b. DB2-TDSV63DB2 - DB2TDS63-0 (Content Platform Engine DB2 instance)

   c. IBM Tivoli Directory Server Instance V6.3 - dsrdbm01 (Directory server)

2. Start WebSphere Application Server.

APPENDIX B:Debugging and troubleshooting

This section describes troubleshooting techniques that can be used when you customize and extend IBM Content Navigator. This section identifies the information that is required and provides instructions to gather it for troubleshooting purpose. Use the information along with IBM Content Navigator documentation for troubleshooting.

Topics

Debugging in the Firefox browser on page 5-6

IBM Content Navigator logging levels on page 5-7

Enable logging in JavaScript files on page 5-8

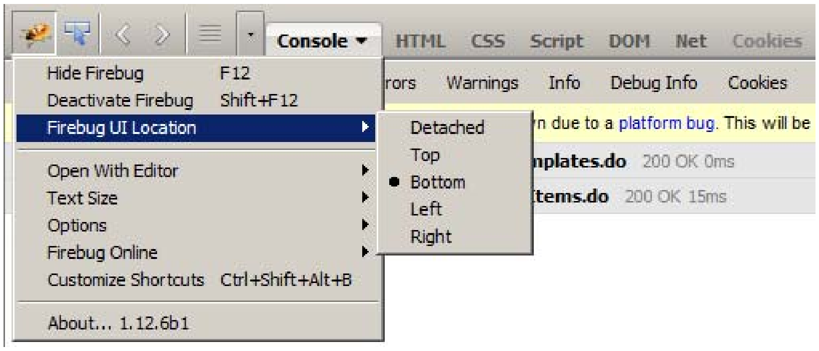
Enabling Server-side logging in IBM Content Navigator on page 5-8

IBM Content Navigator troubleshooting tools on page 5-9

Debugging in the Firefox browser

Firebug (a powerful client debugging tool for Firefox) is installed as a Firefox add-on in your student system.

1. Press F12 to launch the Firebug in the browser.
2. In the Firebug options, it can be set to any location of the browser or separately in an individual window.



3. The following panels in Firebug are useful for different purposes.
   - Console: Show logs and all requests and response content.
   - HTML: Show page HTML source
   - CSS: Show CSS of current page
   - Script: Show scripts. You can debug JavaScript code. You can set breakpoints, watch variable values, and check the whole stack.
   - DOM: Show all dom information.

- Net: Show network transmission information, such as URL, status, size, and timeline.
- Cookie: Show all cookie information.

Note

For more information about Firebug, see: http://addons.mozilla.org/en-US/firefox/addon/firebug

IBM Content Navigator logging levels

IBM Content Navigator provides the `ecm.logger` logging class to use for developing plug-ins.

- The logging class is used to log messages by the following five levels:
  - Level 0: None (This level indicates no logging.)
  - Level 1: Error
  - Level 2: Warning (This level is the default logging level.)
  - Level 3: Information
  - Level 4: Debug
- To invoke the desktop with the debug parameter, use the following line:

  `http://<server_name>:<port>/navigator/?debug=true`

  `<server_name>` is the DNS resolvable name of the web application server that hosts the navigation web client application.

  `<port>` is the port number for the application server. The default for WebSphere Application Server is 9080.
- To view the console in your web browser, select the <F12> function key.

  If you use Firefox and do not have a debug tool, such as Firebug, installed, the log messages can be redirected to a new page by using the following command:

  `http://<server_name>:<port>/navigator?debug=true&useConsole=false`
- Performance
  - Logging debug messages affects performance. However, newer versions of browsers are more efficient and has less effect on load times.
  - Using the debug=true parameter is the equivalent of setting the logging level to 4. If you want to reduce the logging level, use the logLevel parameter:
  - `http://<server_name>:<port>/navigator?logLevel=1&useConsole=false`
  - The `useConsole=false` parameter logs the errors to a new page.

Enable logging in JavaScript files

- To enable the Enterprise Content Manager (ECM) logger in your JavaScript, include the following statement:

  `dojo.require(“ecm.LoggerMixin”);`
- The ecm.LoggerMixin class provides an ECM logger class and provides functions for logging the various level messages.
  - To log a message with one of the functions that the LoggerMixin class provides, use the following format:

  `this.logInfo(yourfunctionname, message, extra);`
- LoggerMixin provides the following functions:
  - logInfo: Log an information message.
  - logWarning: Log a warning message.
  - logError: Log an error message.
  - logDebug: Log a debug message
- Two more message types can be used:
  - `logEntry (Function Name, Message)`: Log the entry point to your function.
  - `logExit (Function Name, Message)`: Log the exit point from your function.

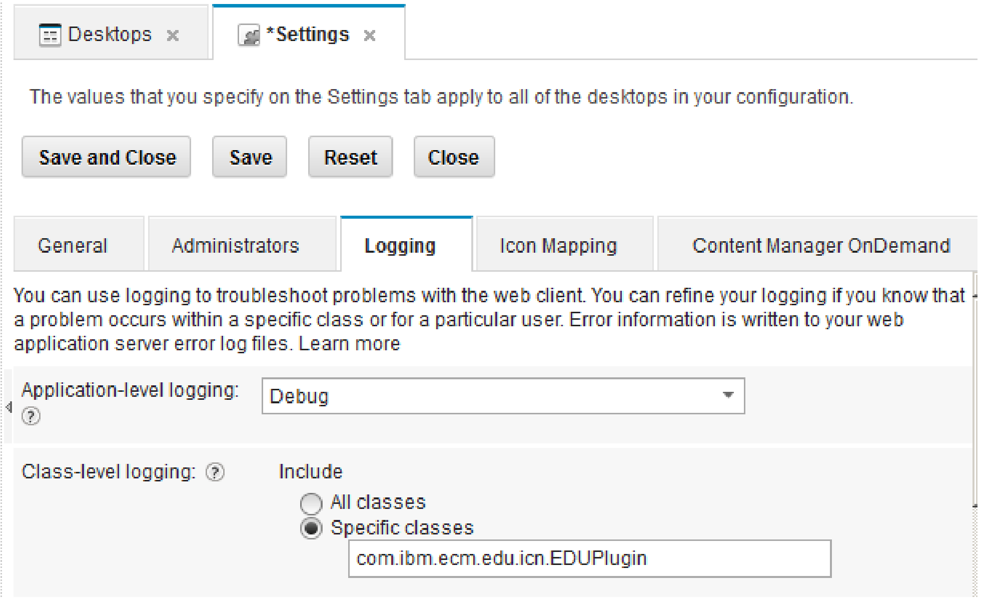
Enabling Server-side logging in IBM Content Navigator

Logging can be enabled for specific classes. Use this option to debug your classes during development. You can optionally exclude specific classes.

In this section, you use EDUPlugin as an example plug-in for the server logging to trace the execution of the main Java class.

1. In your Firefox browser, go to `http://ecmedu01:9080/navigator/?desktop=admin`.
2. Enter the logon credentials for an administrator.
   - User ID: `p8admin`
   - Password: `IBMFileNetP8`
3. In the Admin desktop, click the Settings icon in the left pane.
4. In the Settings tab on the right pane, select the Logging subtab.
   a. Select Debug from the list for the Application-level logging field.
   b. Select Class-level logging > Include > Specific Classes.

- Enter com.ibm.ecm.edu.icn.EDUPlugin.*



The values that you specify on the Settings tab apply to all of the desktops in your configuration.

You can use logging to troubleshoot problems with the web client. You can refine your logging if you know that a problem occurs within a specific class or for a particular user. Error information is written to your web application server error log files. Learn more

Application-level logging: Debug

Class-level logging: Include

○ All classes
● Specific classes

com.ibm.ecm.edu.icn.EDUPlugin

- Click Save and Close.

Note

Logging can be enabled for specific users or specific client devices. Use this option if you need to use debug on a production instance of IBM Content Navigator. Restricting the scope of logging reduces the effect on performance and aids troubleshooting and debugging.

5. Server logging is written to the web application server log file. In the student system, the server is WebSphere Application Server. The `SystemOut.log` file, is in the following location: `C:\Program\Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1`

- Logging information is logged as a series of requests that are delimited by the following lines:

  ```
  Begin Request nnn
  End Request nnn
  ```

  nnn is a numeric value, logged in ascending order.

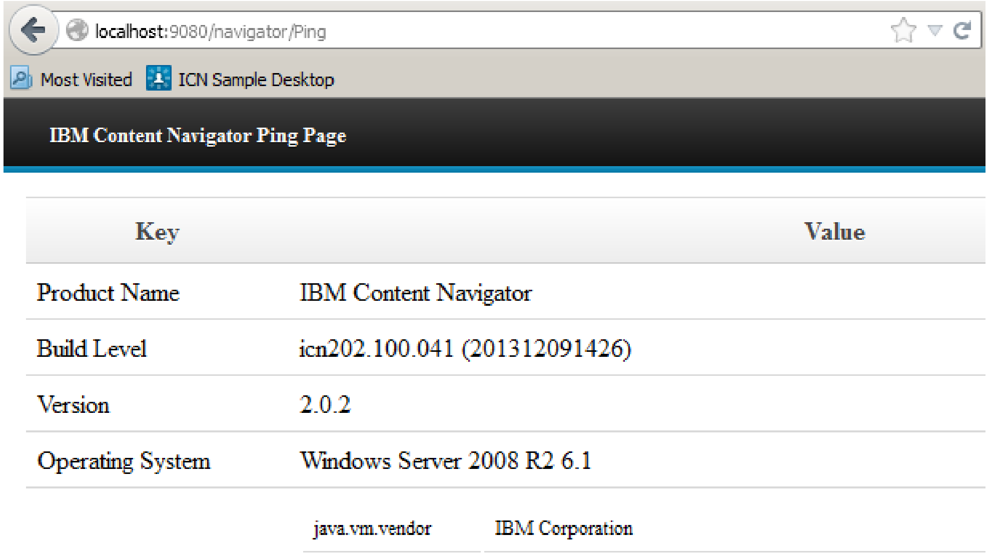
IBM Content Navigator troubleshooting tools

IBM Content Navigator contains the following tools that can help you troubleshoot:

IBM Content Navigator Ping Page

IBM Content Navigator contains a ping page that shows information about the installed software and the status.

- To go to the ping page, enter the following URL in your web browser: http://<server_name>:<port>/navigator/Ping



IBM Content Navigator Ping Page

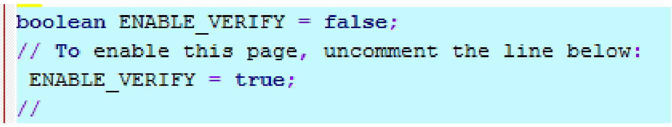
Key	Value
Product Name	IBM Content Navigator
Build Level	icn202.100.041 (201312091426)
Version	2.0.2
Operating System	Windows Server 2008 R2 6.1

java.vm.vendor	IBM Corporation

Viewer Configuration Verify

IBM Content Navigator provides a rich set of viewing capability.

- To check the configuration of the various viewing options, edit the following file in the Navigator deployment directory to remove the comment from the following line:



```
boolean ENABLE_VERIFY = false;
// To enable this page, uncomment the line below:
 ENABLE_VERIFY = true;
//
```

- Enter the following URL in your web browser: http://<server_name>:<port>/navigator/viewers/verify.jsp
- The verify.jsp page checks the presence and validity of the IBM Content Navigator views that are installed on the system.

APPENDIX C:Develop ECM widgets solutions in IBM Content Navigator

Enterprise Content Management (ECM) Widgets

Enterprise Content Manager widgets provide user interface components for building IBM FileNet Business Process Management (BPM) applications.

ECM widgets are based on the iWidget specification and hosted in Business Space powered by WebSphere.

developerWorks article

The developerWorks article:

- Provides guidance for developing applications with IBM Content Navigator and compares to how applications are developed in Enterprise Content Management widgets.
- Compares the BPM features in each offering, describes the differences in the user interfaces, and describes how applications are customized within each environment.
- Includes a sample that illustrates how to build a custom IBM Content Navigator step processor that is similar to an ECM widgets step processor.
- Applies to IBM Content Navigator 2.0.1 and ECM widgets 4.5.2.

For more information, see the developerWorks article at this URL:

https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=e8206aad-10e2-4c49-b00c-fee572815374#fullpageWidgetId=Wf2c4e43b120c_4ac7_80ae_2695b8e6d46d&file=18d04563-2932-47d7-a559-cc19f6925dc5

APPENDIX D:Additional information and links

IBM Content Navigator Redbook

- IBM Content Navigator Redbooks publication guides you through the Content Navigator platform, its architecture, and the available programming interfaces.
  - Title: Customizing and Extending IBM Content Navigator
  - URL for download: ftp://www.redbooks.ibm.com/redbooks/SG248055/
- The book shows how you can:
  - Configure and customize the user interface with the administration tools provided
  - Customize and extend Content Navigator with the available development options with sample code.
    - Set up a development environment
    - Develop plug-ins that add new action, service, and feature to the user interface.
    - Implementing request and response filters and external data services (EDS)
    - Creating custom step processors
    - Use Content Navigator widgets in other applications
    - Develop Mobile application
    - Customize a Viewer
    - Component deployment and debugging and troubleshooting.
- The book is intended for IT architects, application designers, and developers who works with IBM Content Navigator and IBM ECM products.
- It offers both a high-level description of how to extend and customize IBM Content Navigator and also more technical details of how to do implementation with sample code.

Links to Information Centers and Resources

- IBM FileNet P8 Version 5.2 Information Center
  - http://pic.dhe.ibm.com/infocenter/p8docs/v5r2m0/index.jsp
- IBM Content Management Information Center (with IBM Content Navigator)
  - http://pic.dhe.ibm.com/infocenter/cmgmt/v8r5m0/index.jsp
- IBM Content Manager OnDemand Information Center (with IBM Content Navigator)
  - http://pic.dhe.ibm.com/infocenter/cmod/v9r0m0/index.jsp
- IBM DeveloperWorks forums
  - http://www.ibm.com/developerworks/forums/forum.jspa?forumID=2868&cat=19
- IBM Developer Works Technical Library
  - www.ibm.com/developerworks/library/

- IBM Redbooks publication: Customizing and Extending IBM Content Navigator
  - ftp://www.redbooks.ibm.com/redbooks/SG248055/
  - http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248055.html?Open
- Dojo Mobile
  - http://dojotoolkit.org/features/mobile
  - http://dojotoolkit.org/reference-guide/1.9/dojox/mobile.html

IBM®