Customizing and extending IBM Content Navigator

Introduction

Welcome to the Hands-on Lab for IBM Content Navigator 2.0.

This lab is intended for individuals that have a basic understanding of the end-user runtime feature of IBM Content Navigator and are now interested in learning more about customizing and extending the product to better fit their organization's environment. It provides attendees with an overview of many of the key administration features and extension points of IBM Content Navigator.

As this lab is being run from a VMware image on a notebook computer, performance will not be equivalent to the performance obtained by running IBM Content Navigator in an enterprise configuration.

Content Navigator Administration

This chapter introduces you to the Content Navigator administration.
The Content Navigator administration is used to manage all Content Navigator features, including managing the connections to your enterprise ECM repositories, role specific views called desktops, menus, and labels.

Step 1 – Login to Windows

To begin, log in at the Windows operating system level using the **Administrator** account and the password **FileNetP8123**. To keep things simple for you, all other passwords on all levels are **IBMFileNetP8** in this image.

Step 2 – Start the Content Navigator Administration

Double-click on the Content Navigator Administration Internet Explorer icon on the desktop.

Step 3 – The Administration

You should now be looking at the IBM Content Navigator 2.0 Administration desktop. Login to the application using the following credentials:

User - filenetadmin, Password - IBMFileNetP8

Maximize the VMware image by selecting View > Full Screen from the VMware menu

Maximize the browser to enable it to be viewed more easily.

The Administration interface consists of six major areas:

Desktops

Repositories

Plugins

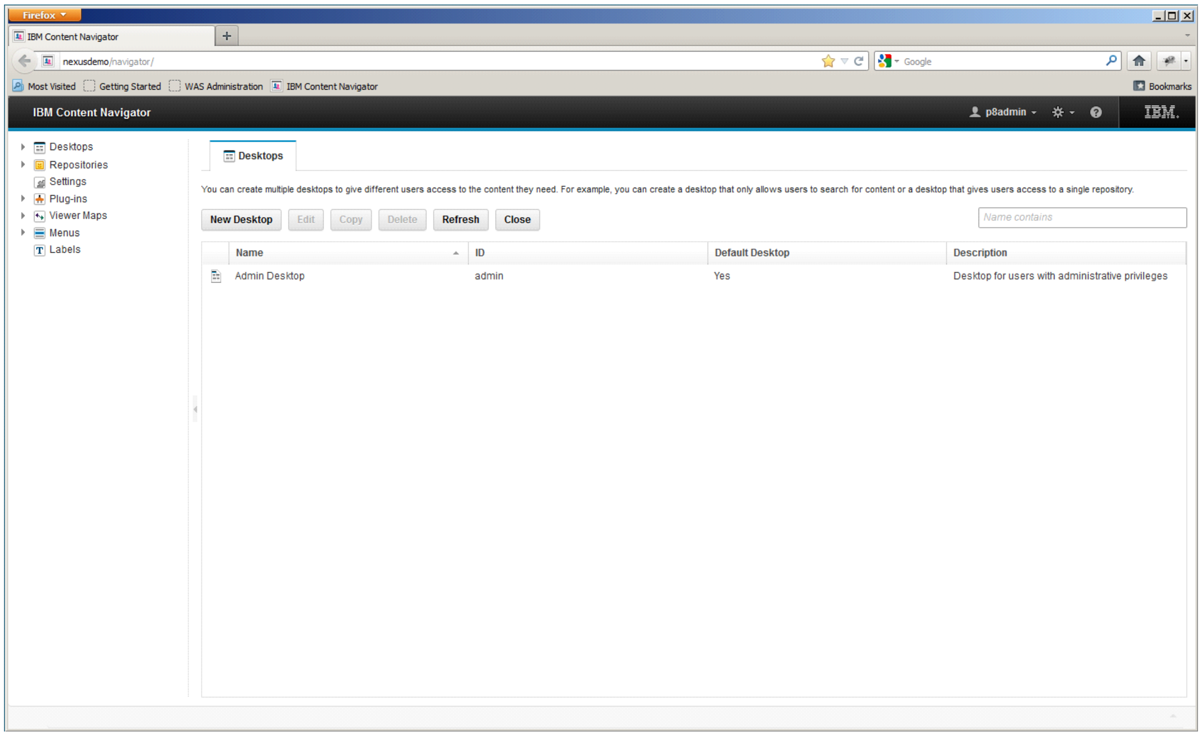
Viewer Maps

Menus

Labels

Each section is reviewed in detail below. Note that the screen shots in this lab manual may not exactly match what you see in your VMware image, but for the purposes of this overview small differences do not matter.

Desktops



You can create multiple desktops to give different users access to the content they need. For example, you can create a desktop that only allows users to search for content or a desktop that gives users access to a single repository.

Name	ID	Default Desktop	Description
Admin Desktop	admin	Yes	Desktop for users with administrative privileges

First, a quick introduction to the concept of Desktops is necessary. You can think of a desktop as a role specific layout. An organization may have many desktops, one for each business unit. Each business unit may have it's own repository instances, unique workflows, and unique use-cases. A desktop allows the administrator to define what the user of that desktop will see when they log in to the application.

The Desktops panel is used to create and manage desktops. A desktop has the following configuration tabs:

General: The General tab is used to specify general settings such as the desktop name, whether the desktop is the overall default, whether users can specify security information, workflow connection point, etc.

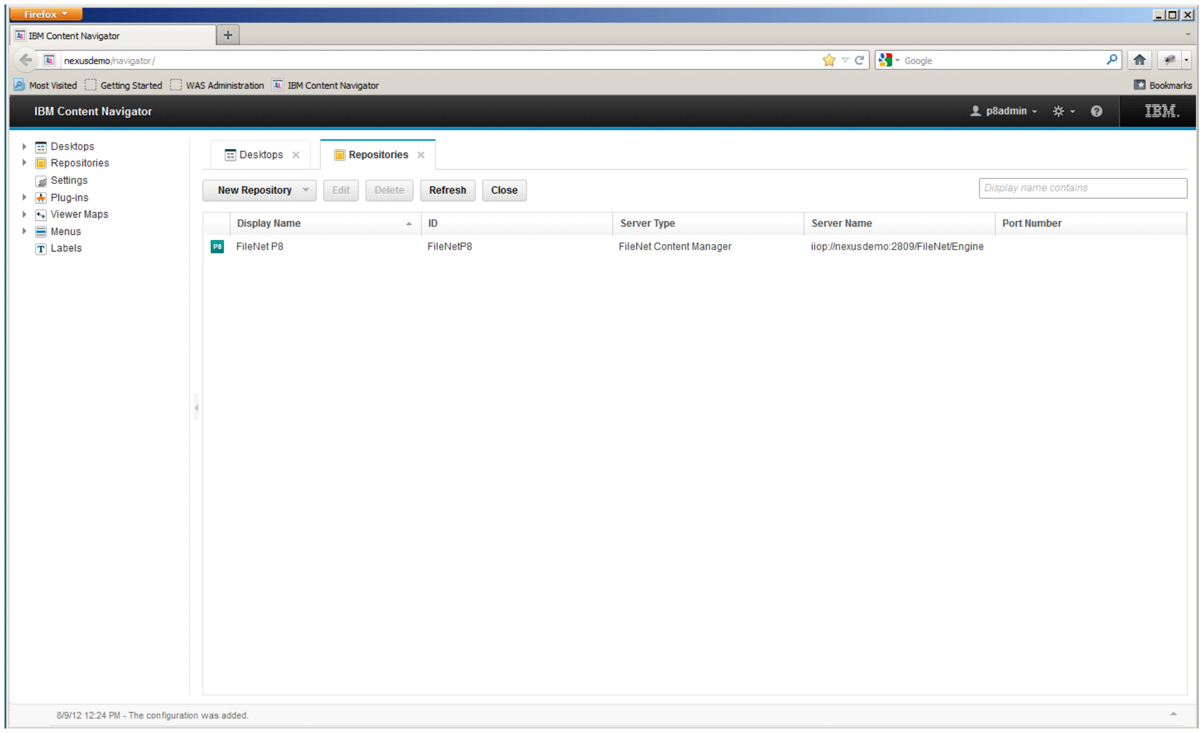
Repositories: The repositories tab is used to select which repositories are assigned to the desktop. This is also where you specify which is the default repository. The default repository is the repository that the user is logging in to when they first access Content Navigator.

Appearance: The appearance tab allows the administrator to customize various visual settings including the application name that will show on the login page and top banner, custom URL's for password rules or help, and what features are available to the desktop. A feature is essentially a section of the application that exposes a particular set of functions. For example, Browse is a feature that has a tree and a content list and allows users to browse repositories. The administrator can select which features are available to the desktop and if there are multiple repositories, which repository is the default repository for certain features such as Search, Browse, or Teamspaces.

Menus: Provides the administrator the ability to customize the menus available to the users of the desktop. If custom menus have been created then they can be selected here and assigned to the desktop.

Workflows: Administrators can assign one or more Application Spaces to the desktop here. This is where you can choose which workflows are assigned to the desktop.

Repositories



Display Name	ID	Server Type	Server Name	Port Number
FileNet P8	FileNetP8	FileNet Content Manager	iiop://nexusdemo:2809/FileNet/Engine	

8/9/12 12:24 PM - The configuration was added.

Repositories are critical to the application as these represent the content management systems to which Content Navigator is connected. Administrators can use this section of the administration to define exactly which repositories are available to the application. After repositories are registered here, they can then be associated with one or more desktops in the Desktops tab we covered in section 2.1.

Content Navigator supports four different types of repositories. Each type of repository has it's own set of tabs for configuring various aspects of the connection to those repositories:

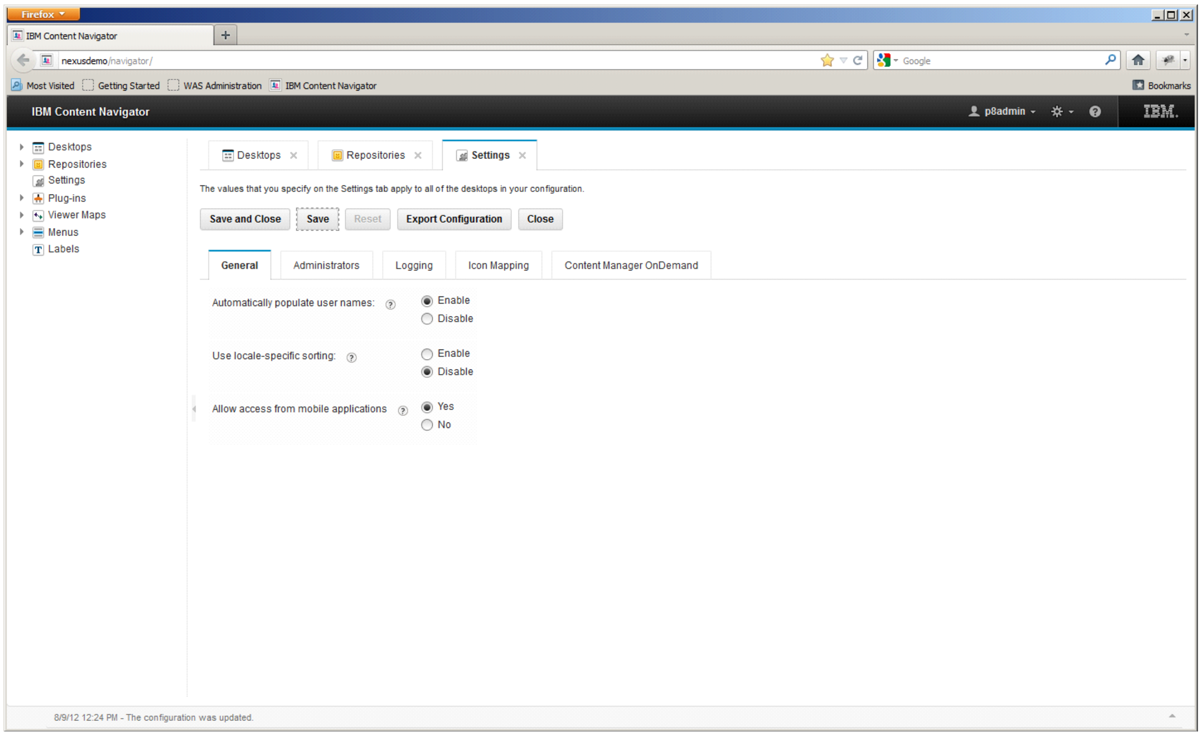
FileNet Content Manager: For FileNet CM repositories, you must create a repository definition for each Object store that you need to access. The General tab allows the administrator to enter basic connection information. After successfully connecting with an administrator account, the other tabs become available. Take a look at each of them and familiarize yourself with the settings.

Content Manager: For Content Manager you create one repository definition for each Library Server.

Content Manager OnDemand: For CMOD, you create one repository for each CMOD Library Server.

Content Management Interoperability Services: Content Navigator V2.0 ships with a technology preview connector to CMIS compliant repositories.

Settings



The values that you specify on the Settings tab apply to all of the desktops in your configuration.

Automatically populate user names: [✓ Enable] Disable

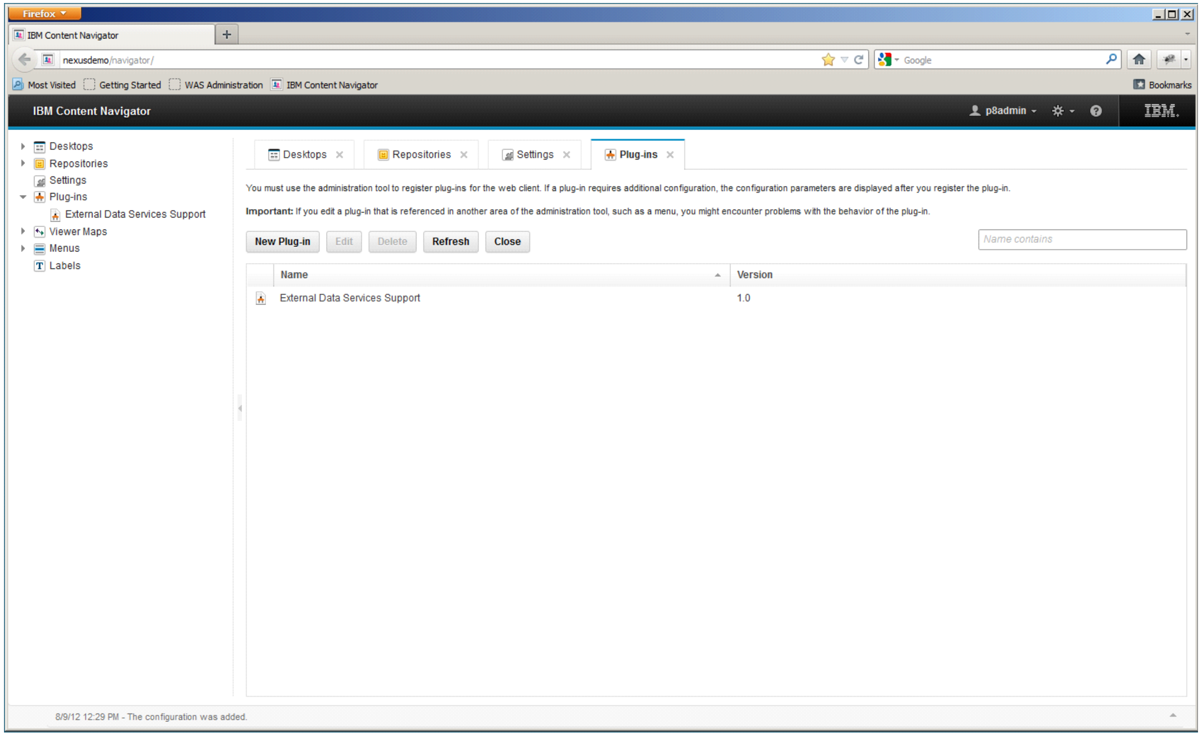
Use locale-specific sorting: Enable [✓ Disable]

Allow access from mobile applications [✓ Yes] No

8/9/12 12:24 PM - The configuration was updated.

The Settings panel is used to control general settings for the entire application, independent of a specific repository or desktop. Take a look at each tab and familiarize yourself with the options available. Of particular note is the switch that allows an organization to disable mobile access from the Content Navigator iPad application and the Icon Mapping tab which allows an organization to override icons used in Content Navigator with their own custom icons. The Icon Mapping is a prime example of how Content Navigator can be customized without writing any actual code.

Plug-ins



You must use the administration tool to register plug-ins for the web client. If a plug-in requires additional configuration, the configuration parameters are displayed after you register the plug-in.

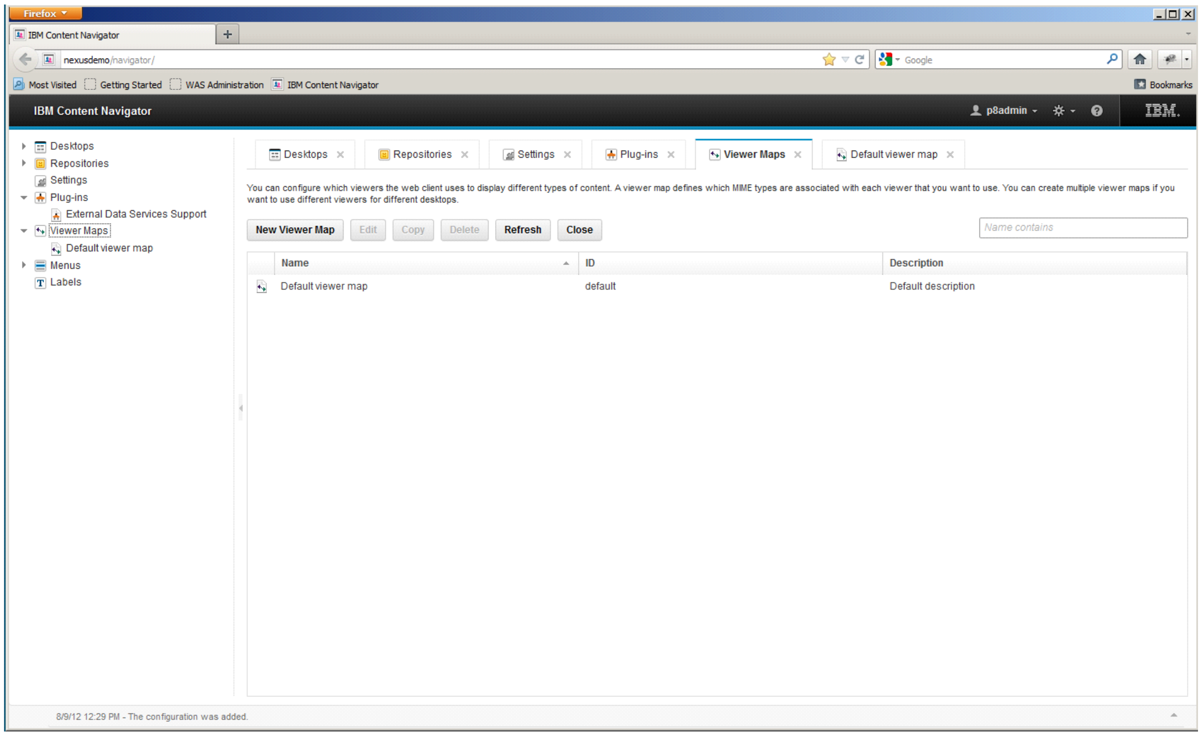
Important: If you edit a plug-in that is referenced in another area of the administration tool, such as a menu, you might encounter problems with the behavior of the plug-in.

Name	Version
External Data Services Support	1.0

8/9/12 12:29 PM - The configuration was added.

The Plug-ins panel is where Administrators can add custom extensions to Content Navigator. We'll discuss Plug-ins in a bit more detail later in this lab.

Viewer Maps



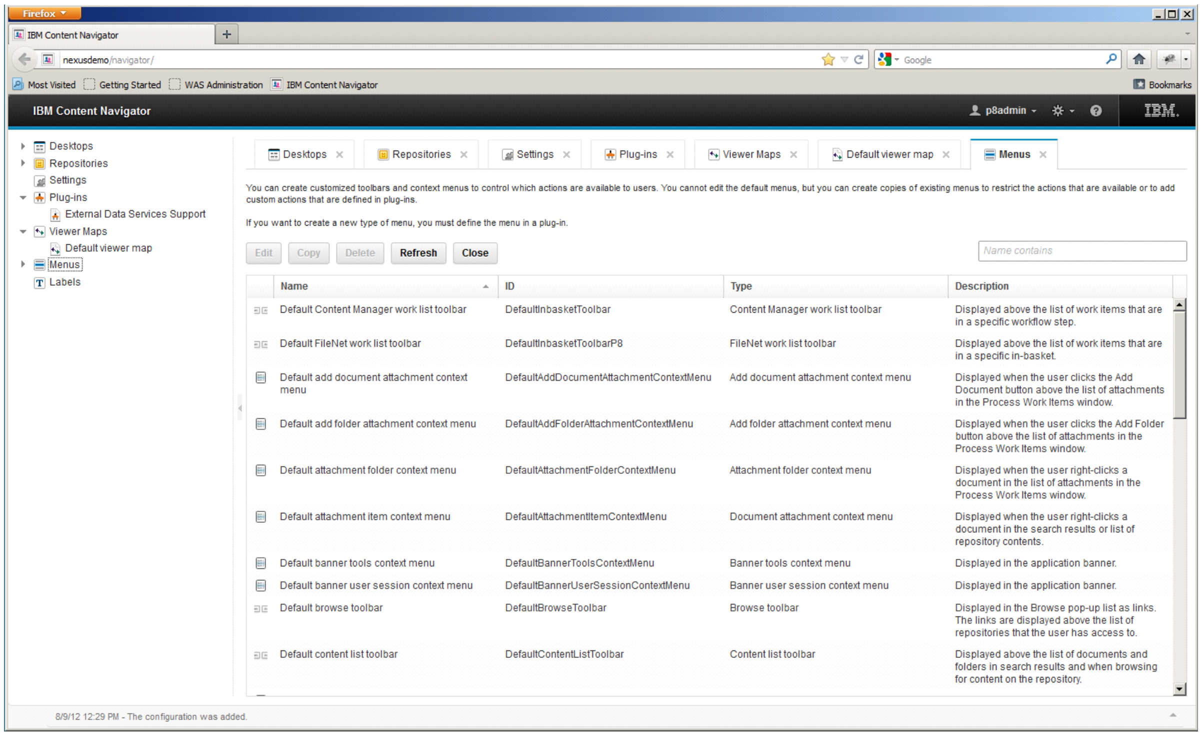
You can configure which viewers the web client uses to display different types of content. A viewer map defines which MIME types are associated with each viewer that you want to use. You can create multiple viewer maps if you want to use different viewers for different desktops.

Name	ID	Description
Default viewer map	default	Default description

8/9/12 12:29 PM - The configuration was added.

The Viewer Maps panel is where Administrators can customize how end-users will interact with the actual content stored in to the content repositories. To be a bit more specific, Viewer Maps control what viewers are launched when a user clicks on a document in any of the various features in Content Navigator. Administrators can specify unique viewers per repository type per mimetype. The viewers are listed in the map in order of precedence so generally the top most entries are the most specific and then the last entries are more general with the intention being that the last entries are the general catch-all setting for any content type not already covered by a viewer definition higher up in the list. This is also where Administrators could register custom viewers that may have been created as a Plug-in by services organizations, or business partners, or even customers themselves.

Menus



You can create customized toolbars and context menus to control which actions are available to users. You cannot edit the default menus, but you can create copies of existing menus to restrict the actions that are available or to add custom actions that are defined in plug-ins.

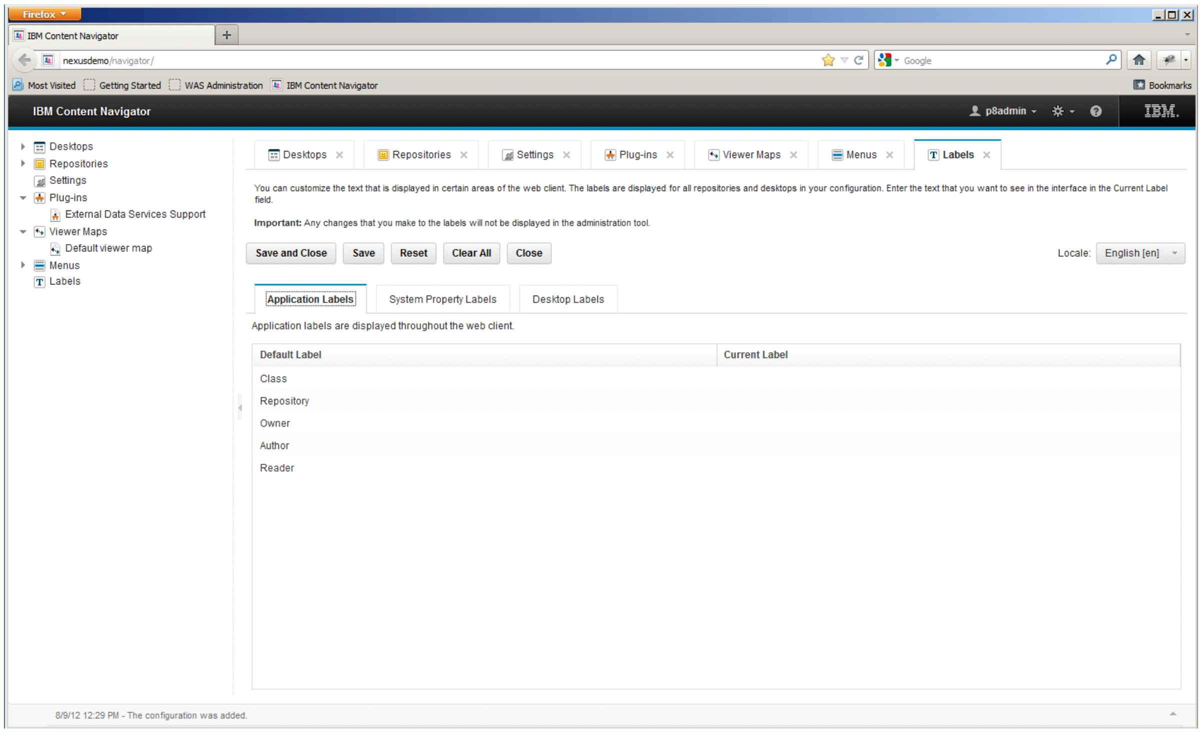
If you want to create a new type of menu, you must define the menu in a plug-in.

Name	ID	Type	Description
Default Content Manager work list toolbar	DefaultInbasketToolbar	Content Manager work list toolbar	Displayed above the list of work items that are in a specific workflow step.
Default FileNet work list toolbar	DefaultInbasketToolbarP8	FileNet work list toolbar	Displayed above the list of work items that are in a specific in-basket.
Default add document attachment context menu	DefaultAddDocumentAttachmentContextMenu	Add document attachment context menu	Displayed when the user clicks the Add Document button above the list of attachments in the Process Work Items window.
Default add folder attachment context menu	DefaultAddFolderAttachmentContextMenu	Add folder attachment context menu	Displayed when the user clicks the Add Folder button above the list of attachments in the Process Work Items window.
Default attachment folder context menu	DefaultAttachmentFolderContextMenu	Attachment folder context menu	Displayed when the user right-clicks a document in the list of attachments in the Process Work Items window.
Default attachment item context menu	DefaultAttachmentItemContextMenu	Document attachment context menu	Displayed when the user right-clicks a document in the search results or list of repository contents.
Default banner tools context menu	DefaultBannerToolsContextMenu	Banner tools context menu	Displayed in the application banner.
Default banner user session context menu	DefaultBannerUserSessionContextMenu	Banner user session context menu	Displayed in the application banner.
Default browse toolbar	DefaultBrowseToolbar	Browse toolbar	Displayed in the Browse pop-up list as links. The links are displayed above the list of repositories that the user has access to.
Default content list toolbar	DefaultContentListToolbar	Content list toolbar	Displayed above the list of documents and folders in search results and when browsing for content on the repository.

8/9/12 12:29 PM - The configuration was added.

The Menus panel is where Administrators can customize the menu definitions for the application. The default menus used by Content Navigator are all registered here and are not editable. If an administrator wishes to override a default menu, then you would simply create a copy of the default menu, rename it to something unique, and then make any changes desired. Once the new menu definition has been saved, it can then be associated with a particular desktop in the Desktops section. Plug-in developers can also create custom menus programmatically that would be registered in the Menus panel after a plug-in has been deployed.

Labels



You can customize the text that is displayed in certain areas of the web client. The labels are displayed for all repositories and desktops in your configuration. Enter the text that you want to see in the interface in the Current Label field.

Important: Any changes that you make to the labels will not be displayed in the administration tool.

Application labels are displayed throughout the web client.

Default Label	Current Label
Class	
Repository	
Owner	
Author	
Reader	

8/9/12 12:29 PM - The configuration was added.

The Labels panel is where Administrators can change many of the default labels used by Content Navigator in the application. For example, you can override application wide labels such as Class to call it Document Type or Object class. Additionally, you can override many System Property labels and you can also override specific labels for each Desktop. Again, this is another example of how Content Navigator can be customized without any actual development effort.

Lab 1 ICN Admin Desktop: Customizing the Desktop

This lab will focus on highlights of the following functionality:

- Configuring menus, views, and other settings
- Desktop creation and configuration

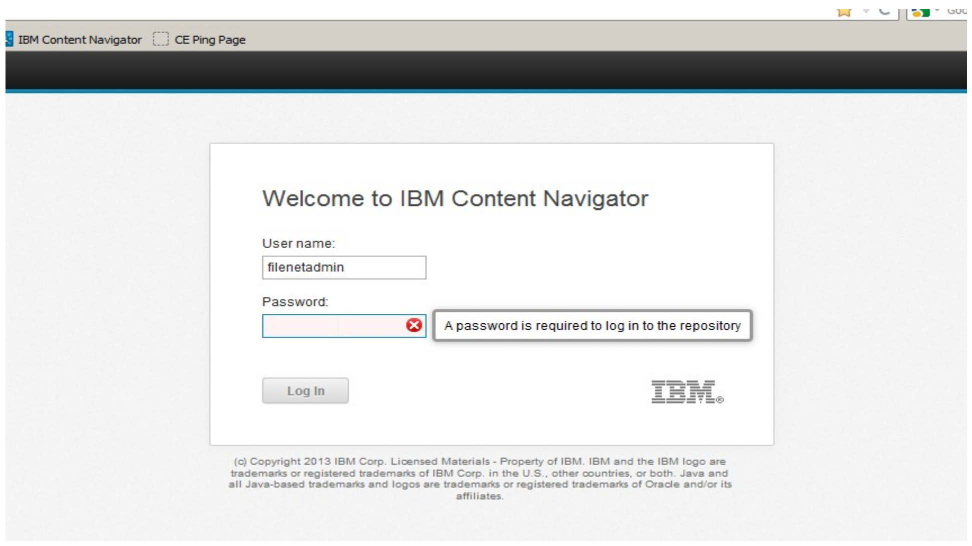
Desktops are the applications presented to users. They define what features are available to which roles, which can include labels, menus, maps, viewers, repositories, and plugins. The Admin Desktop is a special desktop which allows for the configuration of key features. Using web-based tools helps you to respond to changes faster, using secure laptops, mobile, or tablet devices. The ICN Admin Desktop has an intuitive interface and rich functionality to enable you to rapidly respond, customize, and deploy solutions.

Using the Admin Desktop for Easy Configuration

In this section, you'll skim the top of the configurations available in the Admin Desktop. You'll see that very powerful functionality is exposed in a very easy-to-use interface. Using the Admin Desktop, you can quickly customize sources, metadata, functionality, and the look and feel across ICN.

__1. The Admin Desktop is found at http://base-win2k8x64:9080/navigator/?desktop=admin
Open a NEW browser (you should have closed the last one) and navigate to that URL above.

__2. The IBM Content Navigator login page is opened.



Welcome to IBM Content Navigator

User name: filenetadmin

Password:

A password is required to log in to the repository

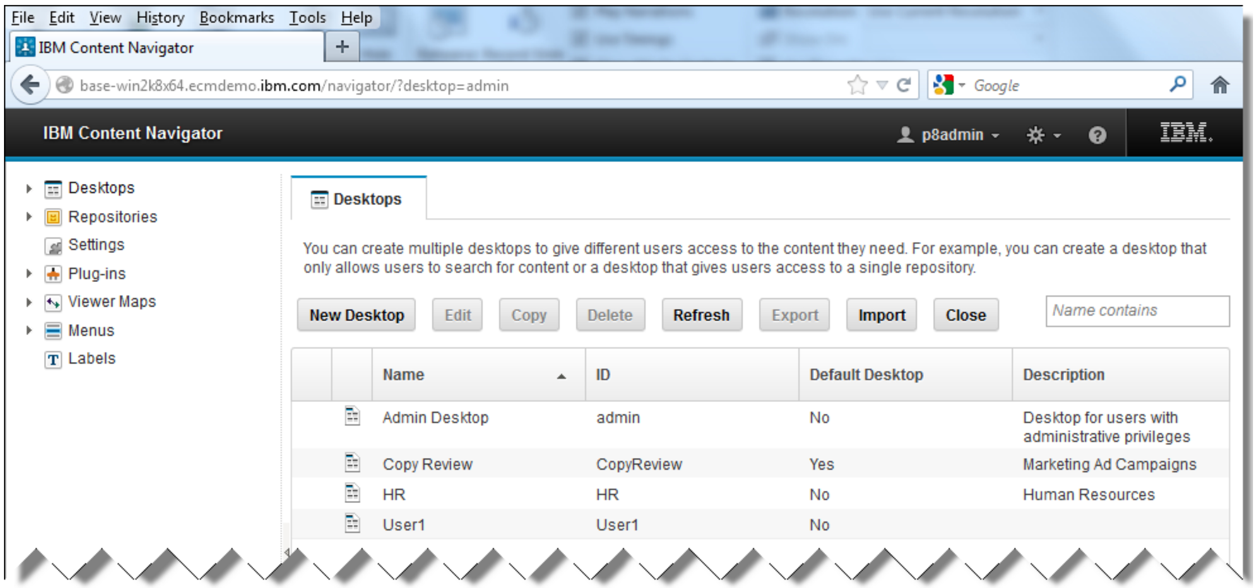
(c) Copyright 2013 IBM Corp. Licensed Materials - Property of IBM. IBM and the IBM logo are trademarks or registered trademarks of IBM Corp. in the U.S., other countries, or both. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

__a. Enter **filenetadmin** for the **User name**.

__b. Enter **IBMFileNetP8** for the **Password**.

__c. Click **Log In**.

__3. The IBM Content Navigator Admin Desktop page is opened. On the left pane are the tools available to the administrator – note how it looks different than the feature panes in a regular desktop.



You can create multiple desktops to give different users access to the content they need. For example, you can create a desktop that only allows users to search for content or a desktop that gives users access to a single repository.

Name	ID	Default Desktop	Description
Admin Desktop	admin	No	Desktop for users with administrative privileges
Copy Review	CopyReview	Yes	Marketing Ad Campaigns
HR	HR	No	Human Resources
User1	User1	No	

__4. The rest of this lab will focus on the repositories and desktops. Exercises following this lab will cover other features.

1.1.1 Repositories, Metadata, and Search

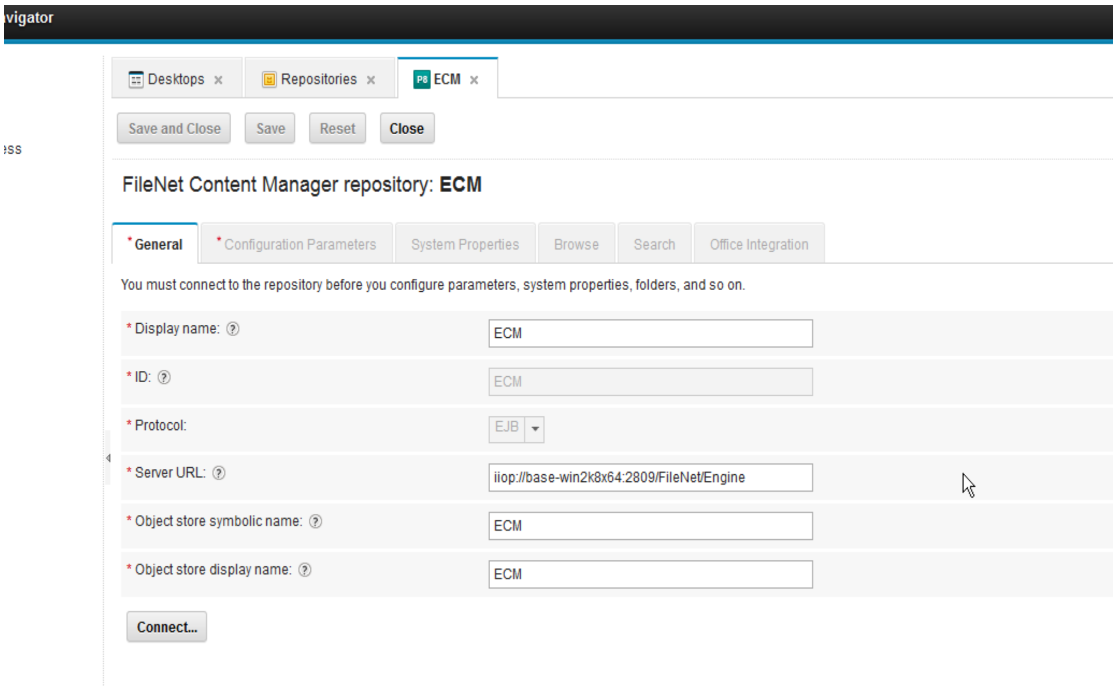
We will look at the settings on a FileNet object store that has been added as another repository to see the configuration. Note: Adding multiple object stores as repositories might not be the best solution for your applications: please consult with an ICN architect.

Repositories, Sessions, and Single Sign On

Keep in mind that if you use multiple object stores, and you switch between the repositories and/or user logins in the browser, you will probably want to implement Single Sign On (SSO) for a cleaner, simpler experience. When switching between users, it is best practice to open a new browser session and login for new users.

In order to configure the repository features that are available, you must first connect to the repository. This will then enable the features supported by your repository.

__5. Click on the **ECM** repository.



FileNet Content Manager repository: **ECM**

You must connect to the repository before you configure parameters, system properties, folders, and so on.

* Display name:	ECM
* ID:	ECM
* Protocol:	EJB
* Server URL:	iiop://base-win2k8x64:2809/FileNet/Engine
* Object store symbolic name:	ECM
* Object store display name:	ECM

Basic connection information and a display name are shown.

__6. Click on the Connect button to get the login prompt and sign in as **filenetadmin** with a password of **IBMFileNetP8**

SYMBOLIC vs. Displayed Name

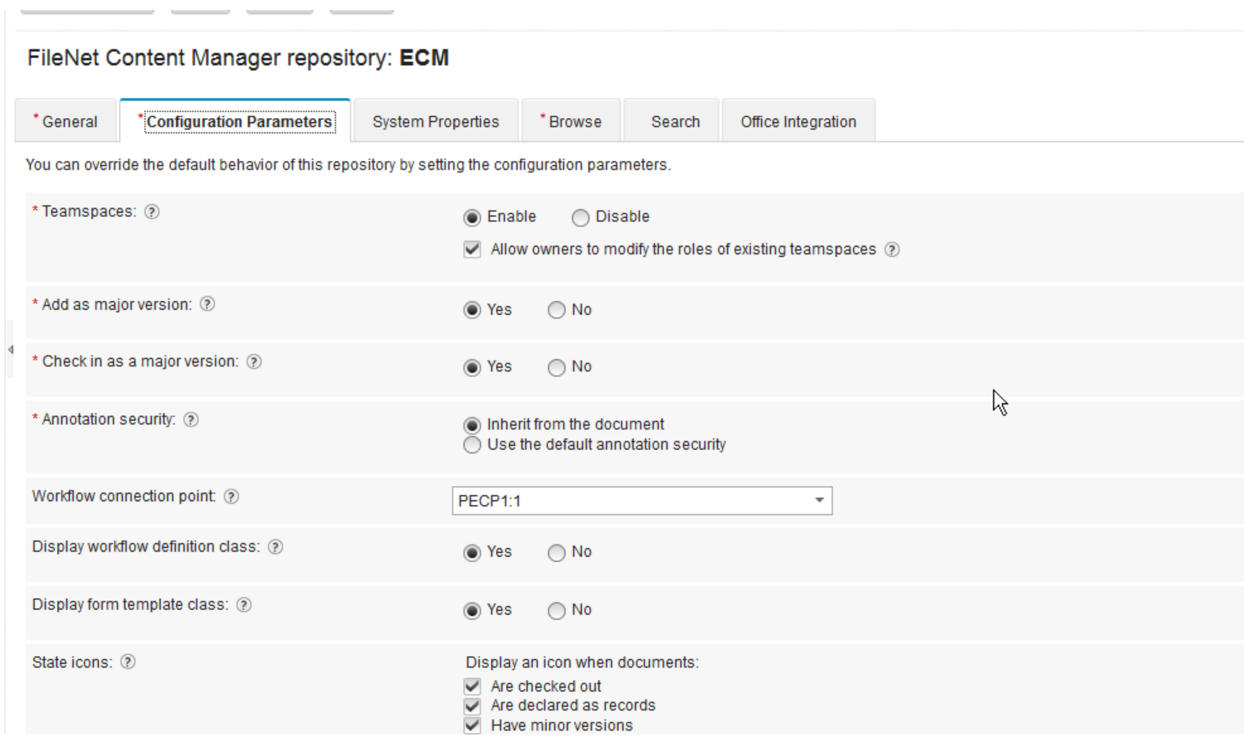
Sometimes the SYMBOLIC name is not the displayed name – if you're having problems, you might take a look in your repository to validate what is correct.

__7. When you have successfully connected, the tabs and features appropriate to this repository will be enabled. Select the **Configuration Parameters** tab.

__8. Here you can see that Teamspaces are enabled, and what the Workflow Connection Point is set to. The Workflow Connection point is needed for work item use. View other settings you can configure for this connection.

Enterprise Records and other plugins

For some plugins like Enterprise Records, you can enable icons to show when something is declared as a record. Enabling special icons to show critical information is a very efficient option.


FileNet Content Manager repository: **ECM**

* General | *Configuration Parameters | System Properties | * Browse | Search | Office Integration

You can override the default behavior of this repository by setting the configuration parameters.

* Teamspaces:	[✓ Enable] Disable ✓ Allow owners to modify the roles of existing teamspaces
* Add as major version:	[✓ Yes] No
* Check in as a major version:	[✓ Yes] No
* Annotation security:	[✓ Inherit from the document] Use the default annotation security
Workflow connection point:	PECP1:1
Display workflow definition class:	[✓ Yes] No
Display form template class:	[✓ Yes] No
State icons:	Display an icon when documents: ✓ Are checked out ✓ Are declared as records ✓ Have minor versions

__9. View the **System Properties** tab by clicking on it. The System Properties tab allows you to customize what metadata is displayed for documents and folders.

__10. Similarly, the **Browse** tab allows you to configure what metadata shows in the content list while browsing, and can be different for different views.

__11. The **Search** tab has important settings that allow you to enable or prevent cross-repository search creation, to limit results and set timeouts, and show and hide properties and thumbnails, among other features.

__12. To configure Microsoft Office Property Mapping, you would select the **Office Integration** tab

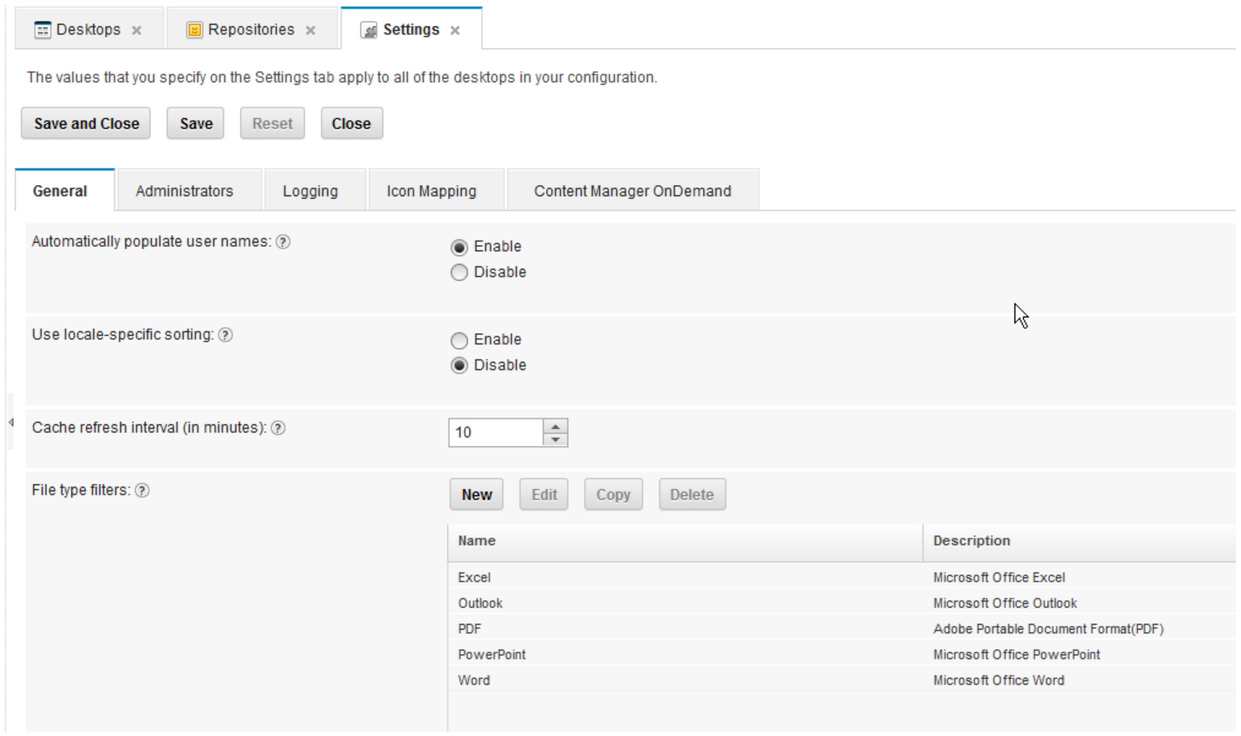
__13. Click the **Close** button. Answer **Yes** to not save any changes you may have made.

__14. Close the **Repository** tab by clicking on the **X** on the tab.

1.1.2 Configuring ICN

The settings panel enables configuration for some key items, such as administrative access, cache timing, and icons. Here you can determine who else can make changes to these specific settings in ICN. Icons can help users to quickly visually determine information, such as urgency, holds, and other key information.

__15. In the Admin Desktop, select the **Settings** tool from the left pane. The Settings tab will appear in the right pane, with several sub-tabs. You will view the features available here.



The values that you specify on the Settings tab apply to all of the desktops in your configuration.

Automatically populate user names: [✓ Enable] Disable

Use locale-specific sorting: Enable [✓ Disable]

Cache refresh interval (in minutes): 10

File type filters:

Name	Description
Excel	Microsoft Office Excel
Outlook	Microsoft Office Outlook
PDF	Adobe Portable Document Format(PDF)
PowerPoint	Microsoft Office PowerPoint
Word	Microsoft Office Word

__16. The **Administrators** sub-tab is used for adding additional admin users for ICN. You may add administrators if you wish.

__17. Select the **Logging** sub-tab. These levels are useful for debugging. Please do not change these settings at this time.

__18. **Icons** can be configured based on state, such as Declared as Record – or something specific to the client solution. You might set icons to show that a document is past due, for instance.

__19. The **Content Manager OnDemand** contains settings that will be used for all CMOD repositories added to ICN.

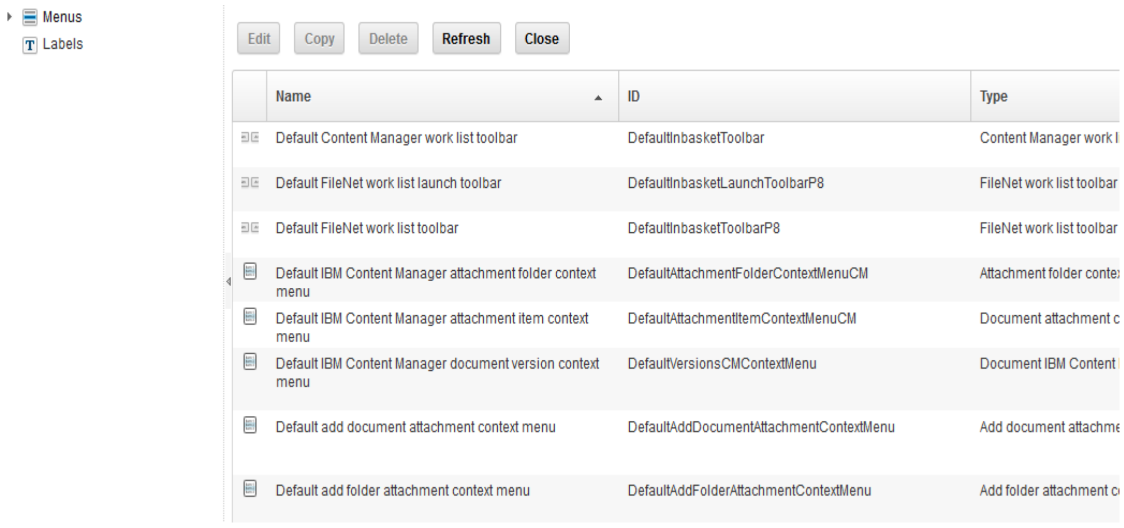
__20. Close the **Settings** tab by clicking on the **X** on the tab.

Customization: Menus and Labels

Customizing menus and labels quickly creates specific solutions for your application. In this lab, you will see how quickly this is accomplished. In this case, we will remove some actions from our users' menus.

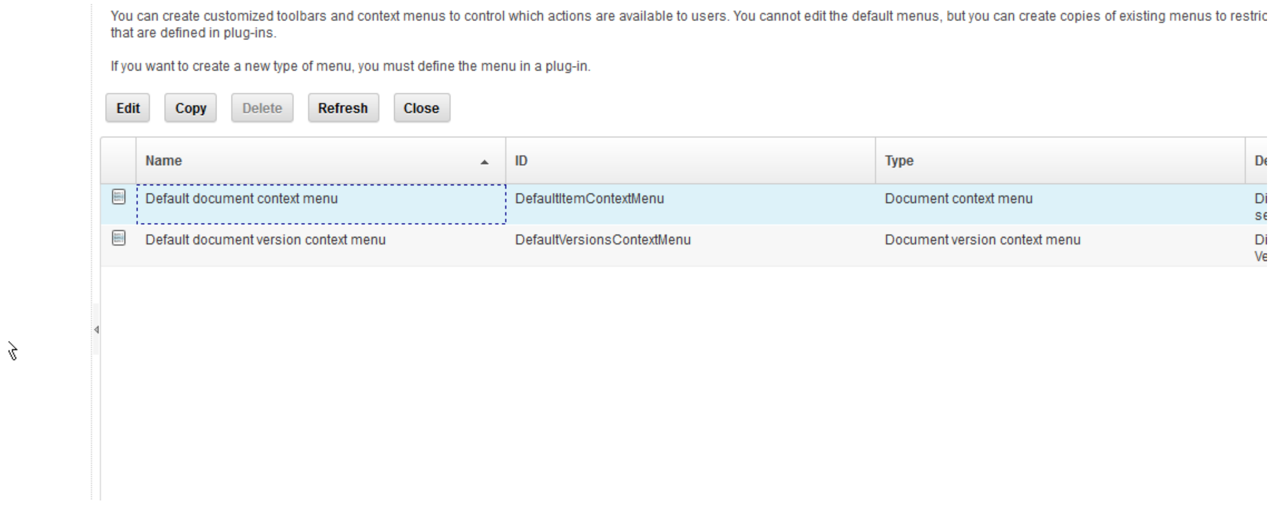
1.1.3 Create a customized menu

1. Click back in your browser, or navigate to URL http://base-win2k8x64:9080/navigator/?desktop=admin
2. Select the **Menus** tool from the left pane. The **Menus** tab will open on the right.



Name	ID	Type
Default Content Manager work list toolbar	DefaultInbasketToolbar	Content Manager work l
Default FileNet work list launch toolbar	DefaultInbasketLaunchToolbarP8	FileNet work list toolbar
Default FileNet work list toolbar	DefaultInbasketToolbarP8	FileNet work list toolbar
Default IBM Content Manager attachment folder context menu	DefaultAttachmentFolderContextMenuCM	Attachment folder conte
Default IBM Content Manager attachment item context menu	DefaultAttachmentItemContextMenuCM	Document attachment c
Default IBM Content Manager document version context menu	DefaultVersionsCMContextMenu	Document IBM Content
Default add document attachment context menu	DefaultAddDocumentAttachmentContextMenu	Add document attachme
Default add folder attachment context menu	DefaultAddFolderAttachmentContextMenu	Add folder attachment c

3. Scroll down the list to find the **Default document context menu (You can also filter by name "Default document context menu" in the text box on the right)**. Select it and the **Copy** button in the menu above the list will become enabled. Click the **Copy** button.

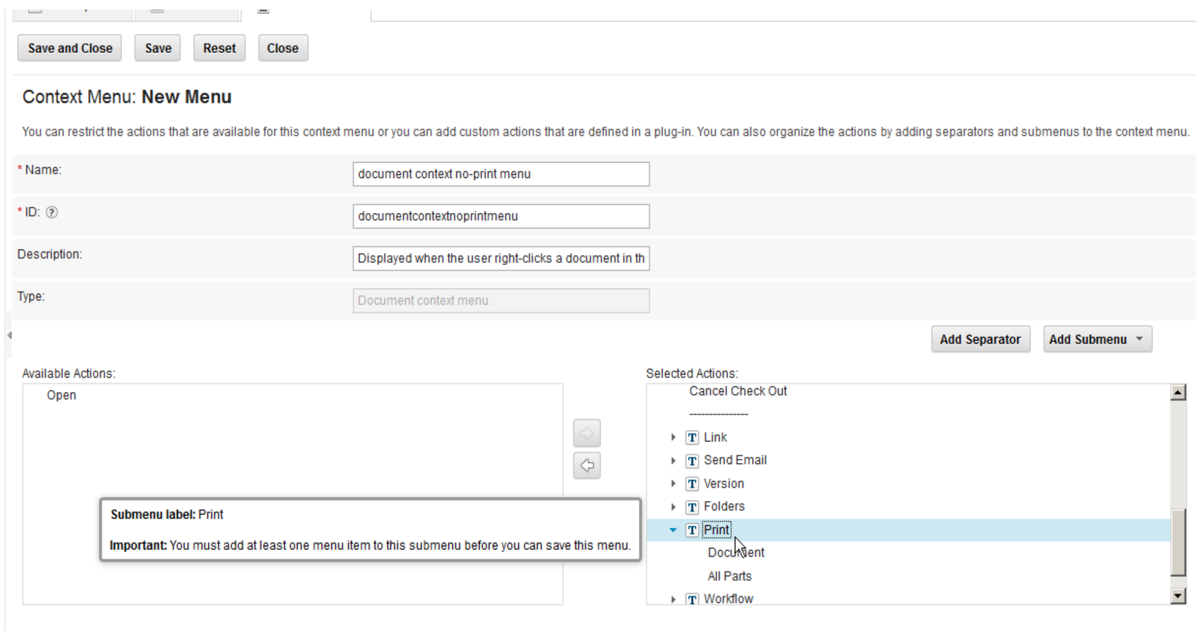

You can create customized toolbars and context menus to control which actions are available to users. You cannot edit the default menus, but you can create copies of existing menus to restric that are defined in plug-ins.

If you want to create a new type of menu, you must define the menu in a plug-in.

Name	ID	Type	De
Default document context menu	DefaultItemContextMenu	Document context menu	Di se
Default document version context menu	DefaultVersionsContextMenu	Document version context menu	Di Ve

The **New Menu** tab opens on the right. Name the new Menu **document context no-print menu.** The ID will fill in by default – this can be altered but it is easier to maintain symbolic and display names that are the same. Leave the description as the default if you like.

4. Remove an action. In the Selected Actions window on the right hand side of the screen, select **Print** and expand the "Print" menu item.



Context Menu: **New Menu**

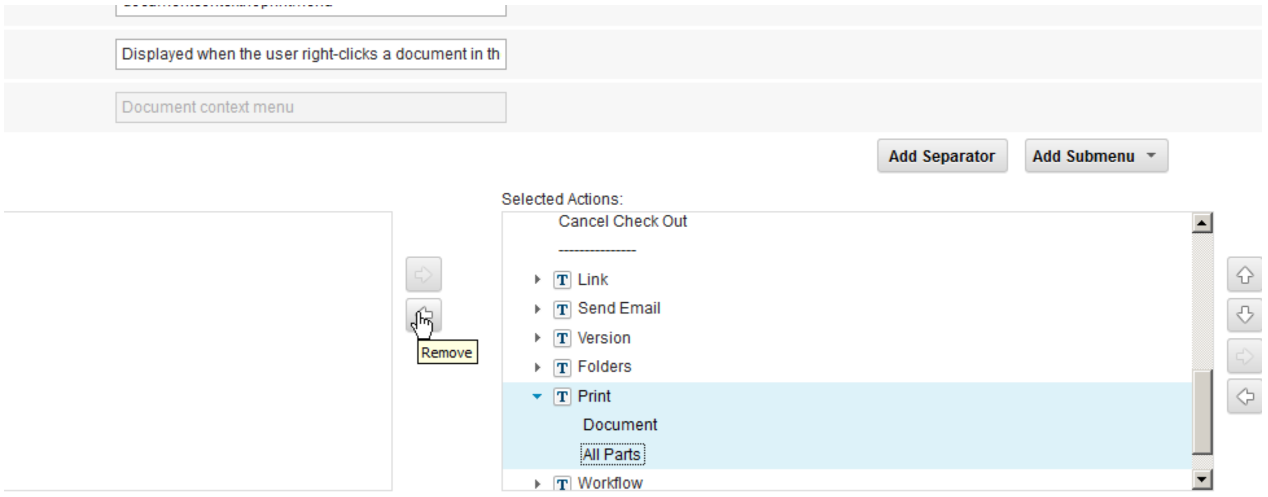
You can restrict the actions that are available for this context menu or you can add custom actions that are defined in a plug-in. You can also organize the actions by adding separators and submenus to the context menu.

* Name:	document context no-print menu
* ID:	documentcontextnoprintmenu
Description:	Displayed when the user right-clicks a document in th
Type:	Document context menu

Submenu label: Print

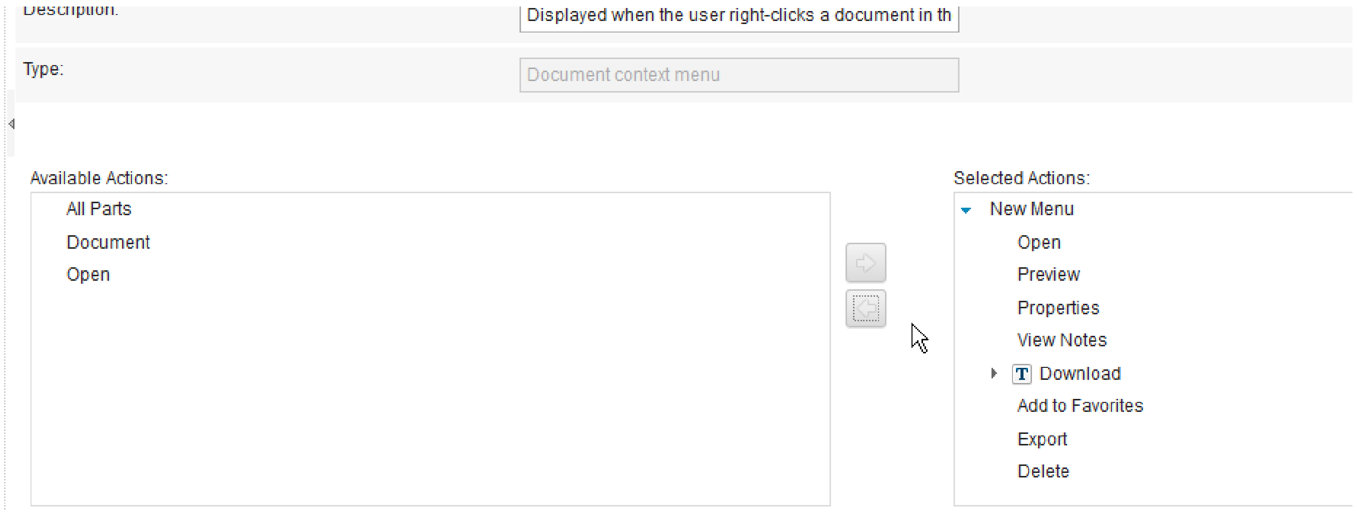
Important: You must add at least one menu item to this submenu before you can save this menu.

5. Next, in the Selected Actions window on the right hand side of the screen, click on the **Document** sub-menu item and while holding down the “Shift” key click on the **All Parts** sub-menu item and click on the “Remove” button.



6. Now you can remove the **Print** Menu item by selecting the **Print** menu item and clicking on the **Remove** button.

7. You should see the “Document” and “All Parts” options in the Available Actions section and the “Print” menu item will be gone from the Selected Actions section.


Description: Displayed when the user right-clicks a document in th

Type: Document context menu

Available Actions:
All Parts
Document
Open

Selected Actions:
New Menu
Open
Preview
Properties
View Notes
Download
Add to Favorites
Export
Delete

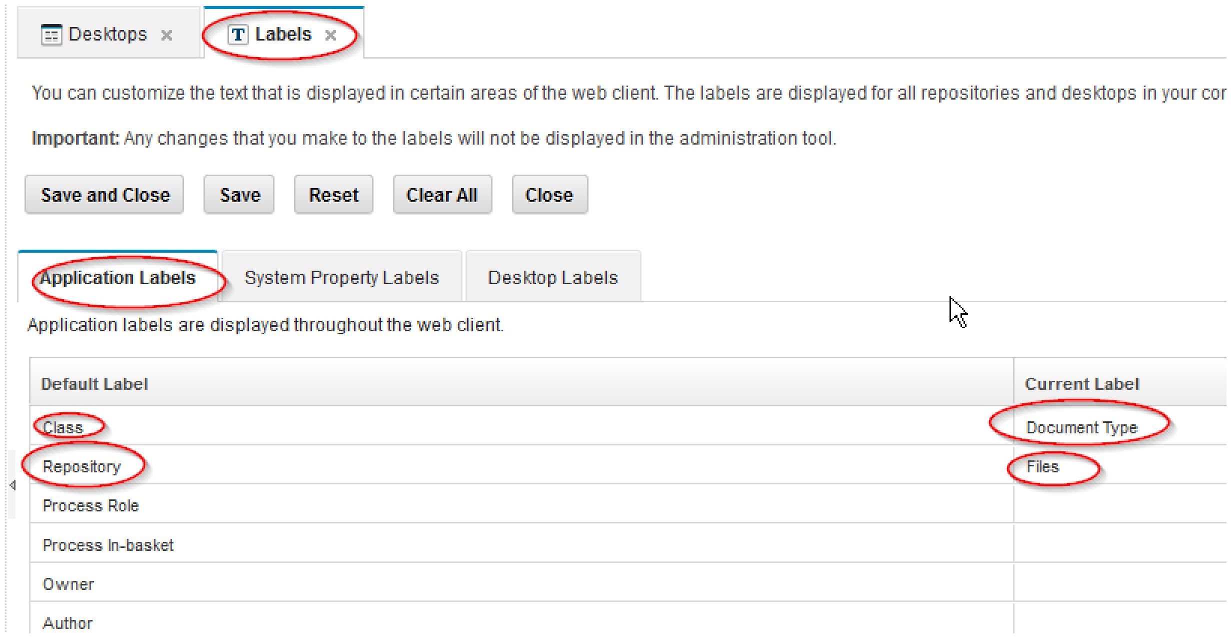
8. Click the **Save and Close** button.
9. Close the **Menus** tab by clicking on the **X** on the tab.

You will use this menu when you create your desktop below.

Create customized labels

This is useful when your users have their own terminology or want things to be more ‘user friendly’. Note that this is mapped for ALL of ICN, not just one desktop or Teamspace.

10. If your browser is not opened to the Admin Desktop, open a browser and navigate to URL http://base-win2k8x64:9080/navigator/?desktop=admin.
11. In the Admin Desktop, go to the **Labels** tool by selecting it on the left pane. The Labels tab will open on the right.
12. On the **Application Labels** sub-tab, select **Class** in the Default Label column. Double-click in the space next to that label in the **Current Label** column.


Desktops ✕ | Labels ✕

You can customize the text that is displayed in certain areas of the web client. The labels are displayed for all repositories and desktops in your cor

Important: Any changes that you make to the labels will not be displayed in the administration tool.

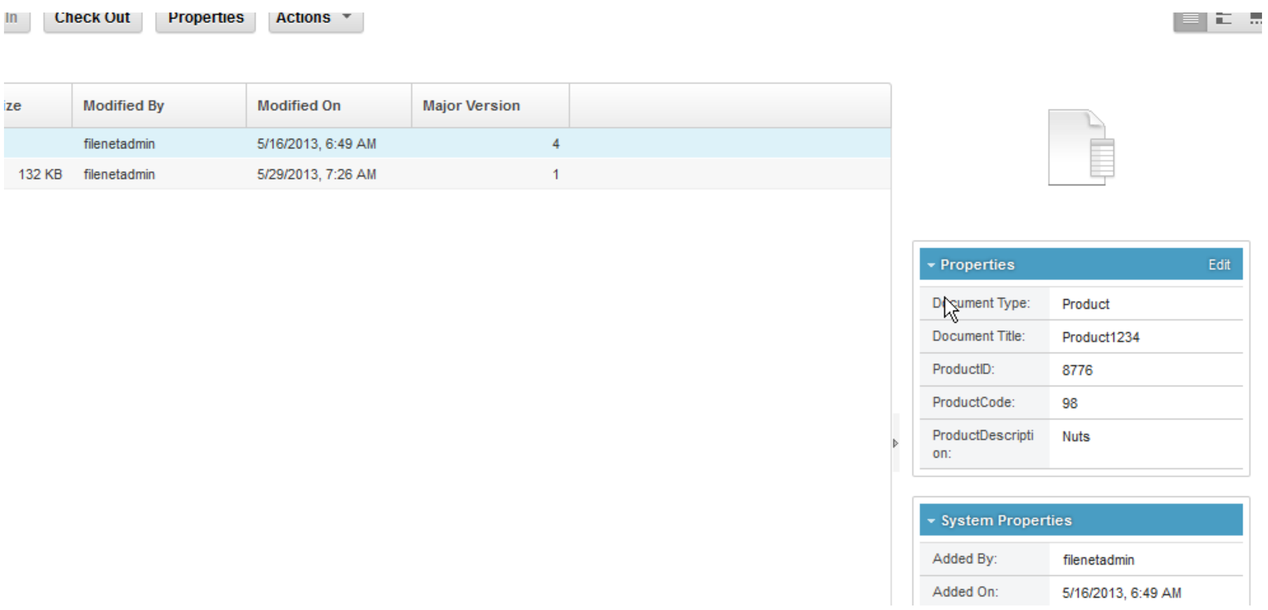
Save and Close | Save | Reset | Clear All | Close

Application Labels | System Property Labels | Desktop Labels

Application labels are displayed throughout the web client.

Default Label	Current Label
Class	Document Type
Repository	Files
Process Role	
Process In-basket	
Owner	
Author	

13. Enter **Document Type** as the new label.
14. Repeat steps and for Repository, this time using a new label of **Files.**
15. Click the Save and Close button.
16. Now, or after the final step of this lab, log out, close the browser, and then reopen it and login to the default desktop to see the change. Select a document and right click on it; those items are renamed – except in the Admin Desktop.
17. Select any document in the list. Notice that the label now says **Document Type** in the Properties section on the right hand side of the Browser. **Add Document** will say **Document Type** also.


In | Check Out | Properties | Actions

ize	Modified By	Modified On	Major Version
	filenetadmin	5/16/2013, 6:49 AM	4
132 KB	filenetadmin	5/29/2013, 7:26 AM	1

Properties Edit

Document Type:	Product
Document Title:	Product1234
ProductID:	8776
ProductCode:	98
ProductDescription:	Nuts

System Properties

Added By:	filenetadmin
Added On:	5/16/2013, 6:49 AM

Summary of Lab 1

As you have seen, it is easy to rapidly develop and customize applications to deliver a custom user desktop using the ICN Admin Desktop.

You have learned

- How to customize the Desktop by adding a new menu
- How to customize the Desktop by adding a new labels

Lab 2 Designing a Semi-Secure Desktop

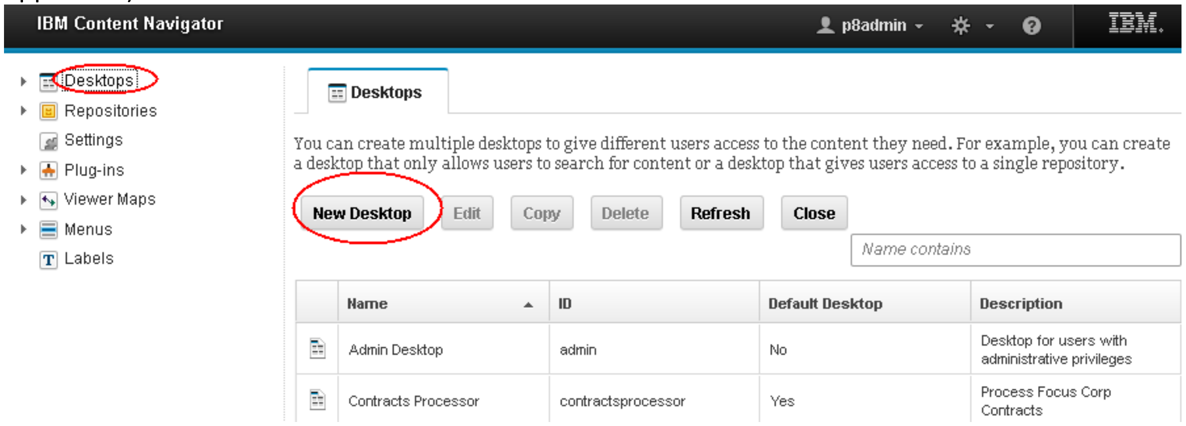
As was mentioned before, desktops are the applications you present to users. They define what features are available to which roles, and can include labels, menus, maps, viewers, repositories, and plugins. The Admin Desktop is a special desktop which allows configurations of key features.

Desktops allow you to quickly create applications with the correct features, sources, metadata, views, and functionality needed for each solution. By using ICN desktops, you'll be able to rapidly roll out custom interfaces VERY quickly, as you'll see in this lab.

Create a New Semi Secure Desktop

In this lab exercise, you will create a new desktop, and configure it with repositories, menus, and other features for your users. You will see how easy it is to create a custom application for your solutions.

__21. In the Admin Desktop, select the **Desktops** tool from the left pane. The Desktops tab will open on the right. (You won't see ContractsProcessor desktop on your image. The screenshot below is not applicable)



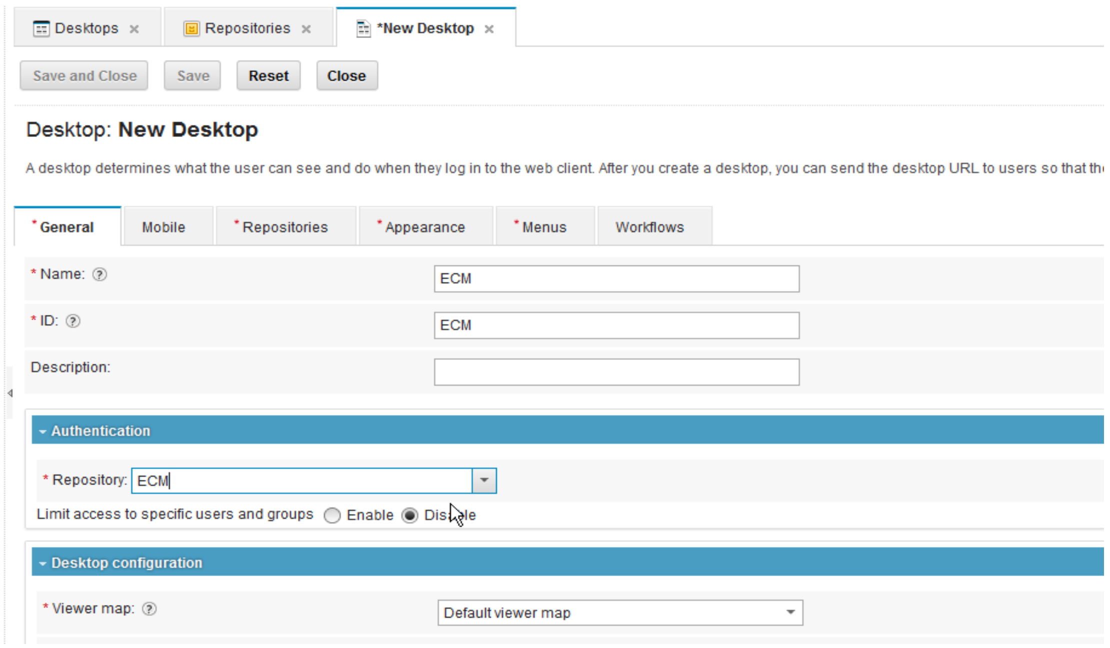
You can create multiple desktops to give different users access to the content they need. For example, you can create a desktop that only allows users to search for content or a desktop that gives users access to a single repository.

	Name	ID	Default Desktop	Description
	Admin Desktop	admin	No	Desktop for users with administrative privileges
	Contracts Processor	contractsprocessor	Yes	Process Focus Corp Contracts

__22. Create a new desktop by clicking the **New Desktop** button. The New Desktop tab will open on the right.

__23. Call it **ECM.** The ID will fill in by default – this can be altered but it is easier to maintain symbolic and display names that are the same. Enter the description "**No Print Options**" and for now select the **ECM** repository.

__24. In the Additional Settings section, check **Require users to save new documents and folders in folders**. Also check **Prevent users from creating searches** and **Prevent users from creating cross-repository searches**.



Desktop: **New Desktop**

A desktop determines what the user can see and do when they log in to the web client. After you create a desktop, you can send the desktop URL to users so that th

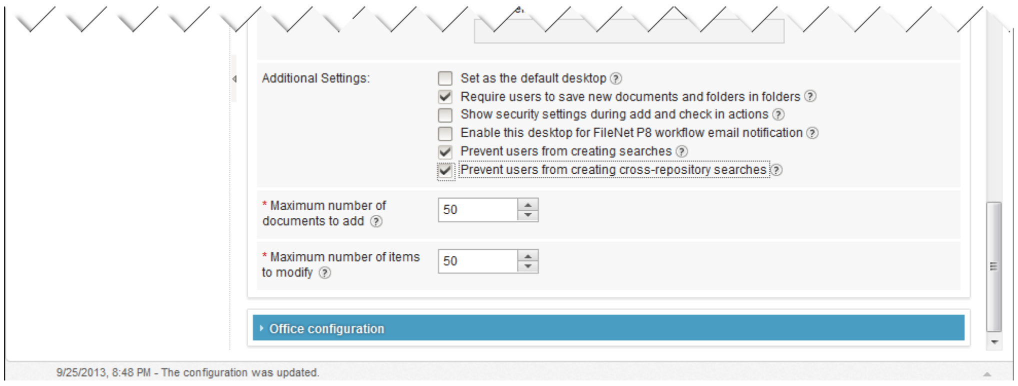
* Name: ECM
* ID: ECM
Description:

Authentication

* Repository: ECM
Limit access to specific users and groups ( ) Enable (●) Disable

Desktop configuration

* Viewer map: Default viewer map


Additional Settings:
- [ ] Set as the default desktop
- [✓] Require users to save new documents and folders in folders
- [ ] Show security settings during add and check in actions
- [ ] Enable this desktop for FileNet P8 workflow email notification
- [✓] Prevent users from creating searches
- [✓] Prevent users from creating cross-repository searches

* Maximum number of documents to add: 50
* Maximum number of items to modify: 50

9/25/2013, 8:48 PM - The configuration was updated.

__25. Select the **Mobile** tab to view the settings. Leave the defaults for **Mobile** (it should be enabled). There will be more on mobile in another lab or demonstration.

__26. Select the **Appearance** tab to do the following customizations (you may choose other colors; these are merely chosen to be very clearly different from the default):

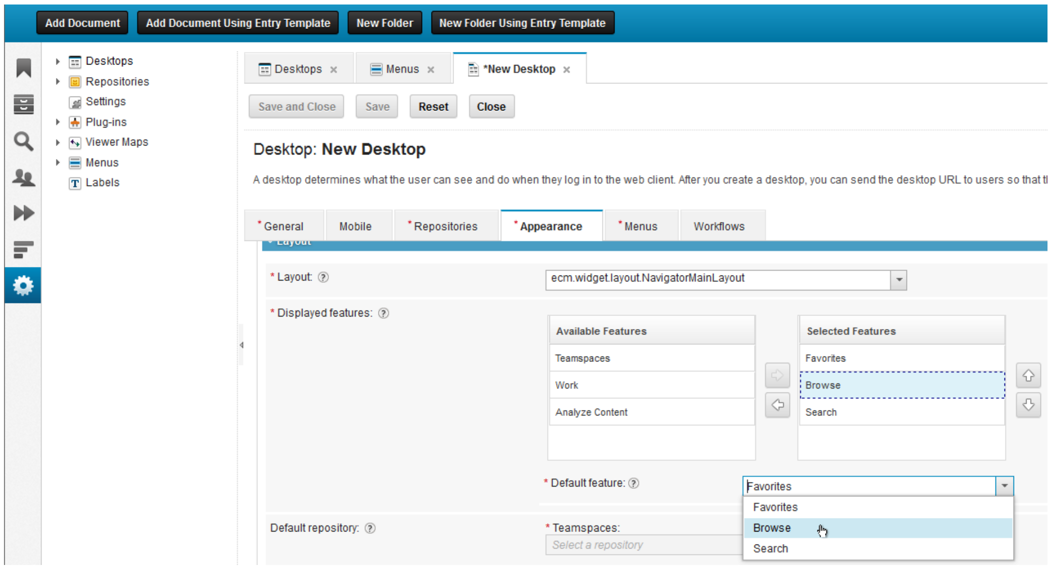
__a. In the **Banner and Login Page** section, enter the following:

__i. Application Name **Enterprise Content Management**

__ii. Choose **Custom** Banner Background color **#FF6600** (zeros)

__b. In the **Layout** section further down the page, configure the following:

__c. Set the **Default feature** to **Browse**.



Desktop: New Desktop

A desktop determines what the user can see and do when they log in to the web client. After you create a desktop, you can send the desktop URL to users so that t

* Layout: ecm.widget.layout.NavigatorMainLayout

* Displayed features:

Available Features	Selected Features
Teamspaces	Favorites
Work	Browse
Analyze Content	Search

* Default feature: Favorites

Default repository:

* Teamspaces: Select a repository

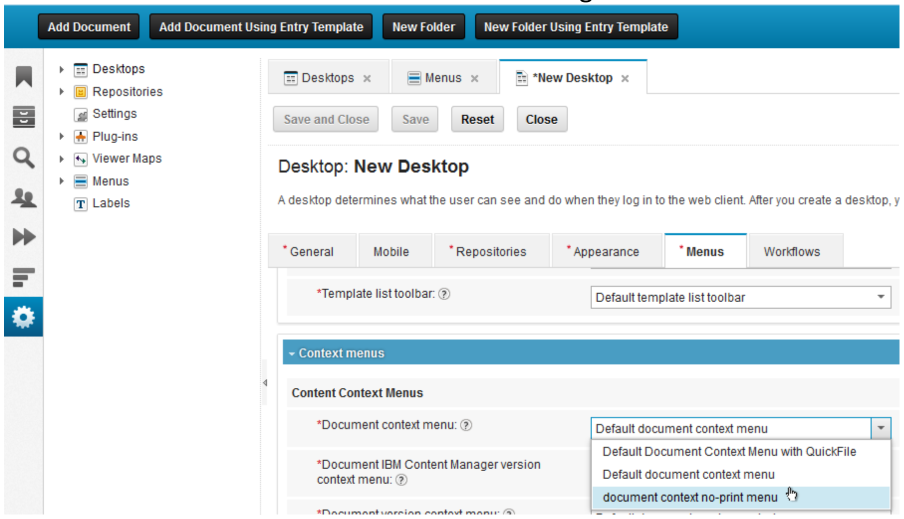
  __i. Set the default repository for Search and **Browse** to **ECM**

__d. Scroll down further and **Enable** document thumbnails. Also **Enable** Filmstrip view.

__27. Click on the **Menus** tab.

__28. In the **Context** menus section (about halfway down the tab) click the drop-down next to the **Document context menu.**

__29. Click on the **Default document context menu** item. Change this to **document context no-print menu**.



Desktop: New Desktop

A desktop determines what the user can see and do when they log in to the web client. After you create a desktop, y

*Template list toolbar: Default template list toolbar

Content Context Menus

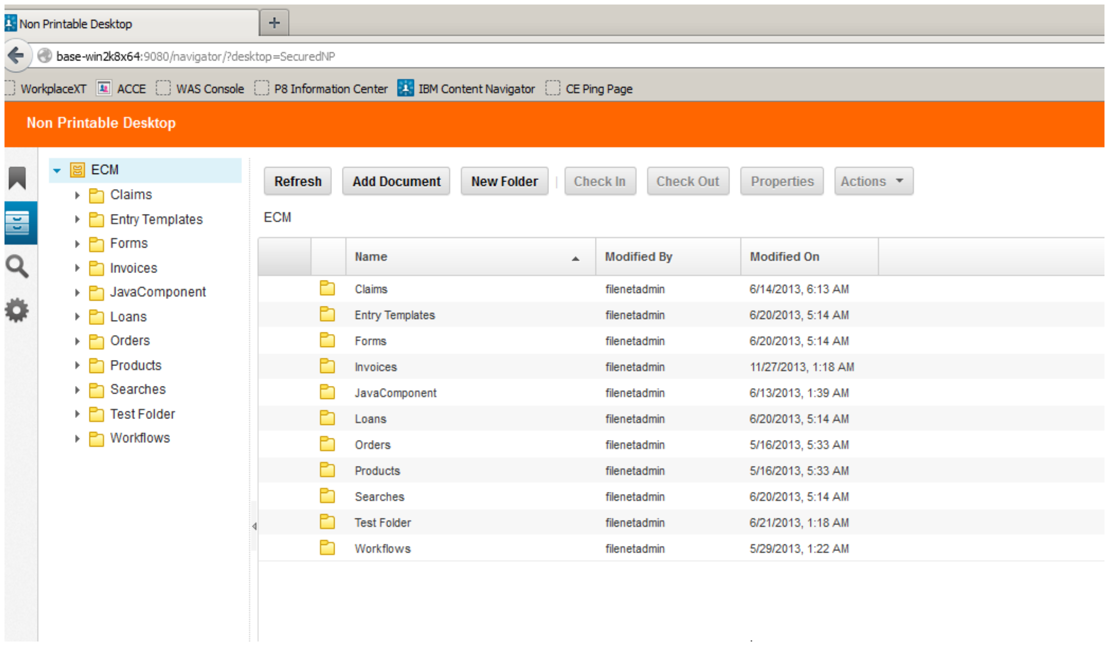
*Document context menu: Default document context menu

*Document IBM Content Manager version context menu:

__30. Click the **Save and Close** button.

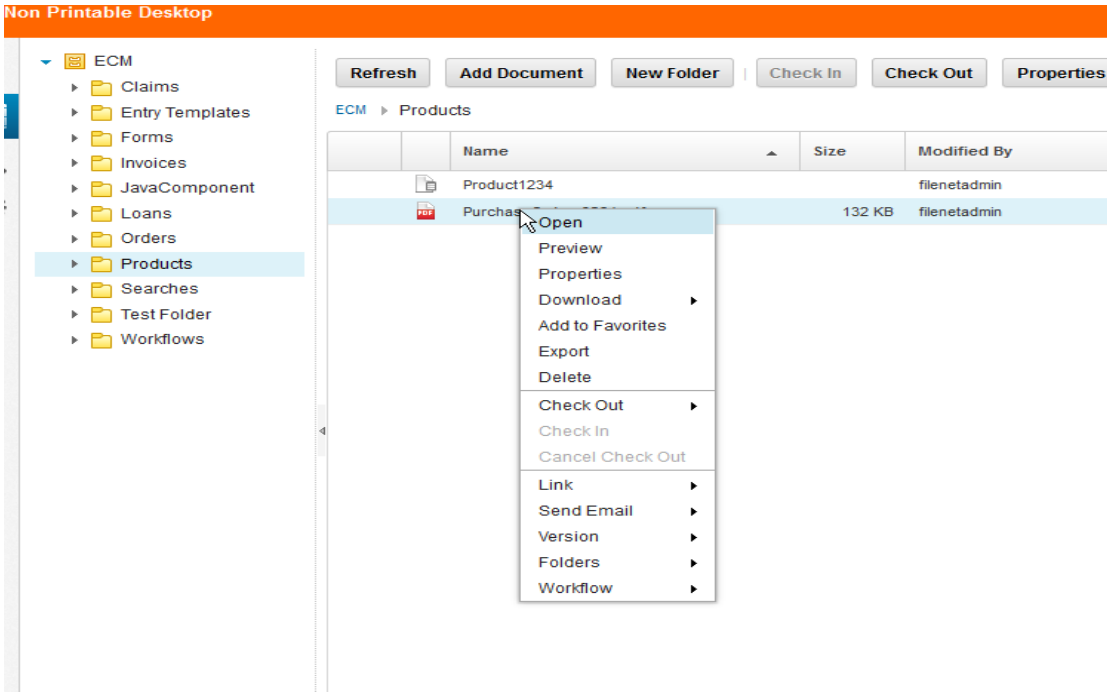
Now that you've created your desktop, try it out!

__31. In your browser's URL, change 'admin' to **ECM** like this: http://base-win2k8x64:9080/navigator/?desktop=ECM. Note that case sensitivity matters. If it looks garish, it's correct. NOTE: In the screenshot below, it shows application name as Non Printable Desktop but you will see application name as Enterprise Content Management



Name	Modified By	Modified On
Claims	filenetadmin	6/14/2013, 6:13 AM
Entry Templates	filenetadmin	6/20/2013, 5:14 AM
Forms	filenetadmin	6/20/2013, 5:14 AM
Invoices	filenetadmin	11/27/2013, 1:18 AM
JavaComponent	filenetadmin	6/13/2013, 1:39 AM
Loans	filenetadmin	6/20/2013, 5:14 AM
Orders	filenetadmin	5/16/2013, 5:33 AM
Products	filenetadmin	5/16/2013, 5:33 AM
Searches	filenetadmin	6/20/2013, 5:14 AM
Test Folder	filenetadmin	6/21/2013, 1:18 AM
Workflows	filenetadmin	5/29/2013, 1:22 AM

__32. You may need to logout, close the browser, then reopen it and login to see some of the other changes in the **ECM** desktop. When you select a document and right click on it, you will notice the print menu item will be gone.



Name	Size	Modified By
Product1234		filenetadmin
Purchas	132 KB	filenetadmin

Note: Use this same sequence when a customer or prospects wants to know how to remove items from the standard document context menu.

Summary of Lab 2

As you have seen, it is easy to rapidly customize desktops to deliver a custom user interface using the ICN Admin Desktop.

You are now familiar with how easy it is to use the Admin Desktop to:

- Rapidly create and distribute custom interfaces for the organization
- Accelerate your ability to respond to business needs
- Remove the ability to print documents from the desktop

Lab 3 Create a Fully Secured Desktop

Using the same techniques we just went through in LAB 1; create a new desktop that only has the **Preview** as content options.
This will show your customers and prospects how they can lock-down the Content Navigator so the users can only Preview n document.
Note: You will need to create a **New Menu** using another copy of the **Default document context menu**.
Call the new menu **Secured document context menu**. Remove all but the **Preview** option.
Now create a new Desktop. Call it **Secured** and only provide the **Favorites** and the **Browse** options.
To further secure this desktop you can log into this new desktop, browse to a specific folder and add that folder to your **Favorites**. Now, you can edit the desktop one more time and remove the Browse feature so the only option a use has is to select the **Favorites** icon.
Without Browse and Search they can only add documents and **Preview** documents within the folder you assigned to the **Favorites**.

Summary of Lab 3

As you have seen, it is easy to rapidly customize desktops to deliver a custom secured user interface using the ICN Admin Desktop.

You are now familiar with how easy it is to use the Admin Desktop to:

- Rapidly create and distribute custom interfaces for the organization
- Accelerate your ability to respond to business needs
- Secure the Desktop from users outputting the content in any way except to preview.

Lab 4: Customization: Menus, Labels, and Viewer Mapping

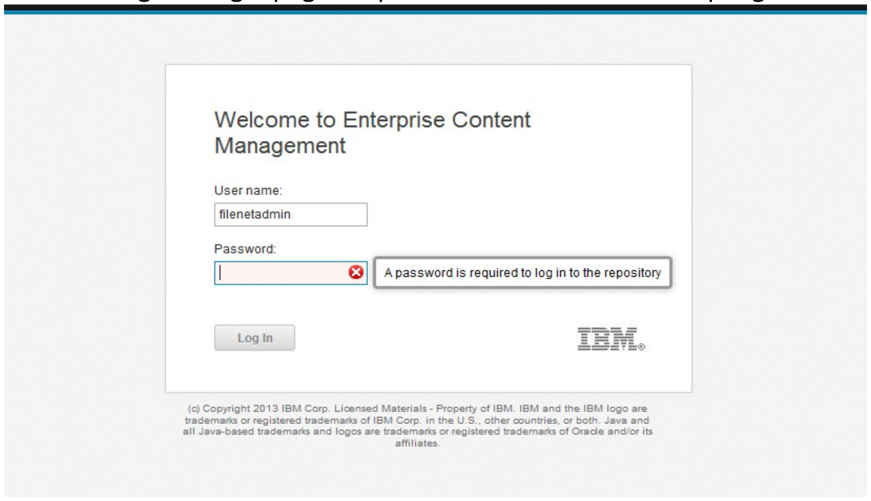
Customizing menus and labels quickly creates specific solutions for your application. In this lab, you will see how quickly this is accomplished. Earlier labs demonstrated how easy it is to customize desktops, searches, and views. This lab touches on a few items that are also useful: Changing the language for a user, creating templates to simplify and standardize Teamspaces, and using different viewers to work on documents.

A. Changing Languages

It is very easy for any user to display the menus and dialogs in their native language.

1. Open the ECM desktop through URL http://base-win2k8x64:9080/navigator?desktop=ECM

The IBM Content Navigator login page is opened with the ECM desktop login showing.



Welcome to Enterprise Content Management

User name: filenetadmin

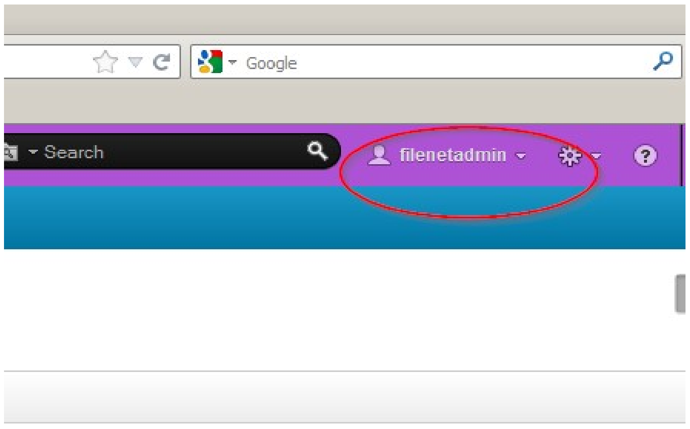
Password:

A password is required to log in to the repository

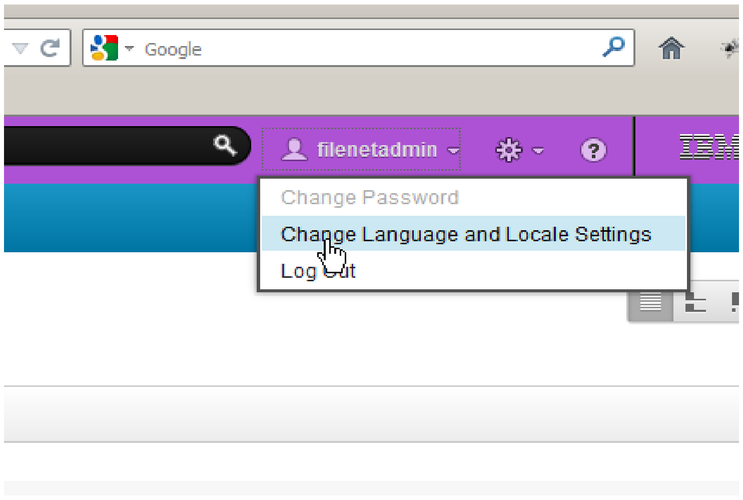
Log In

(c) Copyright 2013 IBM Corp. Licensed Materials - Property of IBM. IBM and the IBM logo are trademarks or registered trademarks of IBM Corp. in the U.S., other countries, or both. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

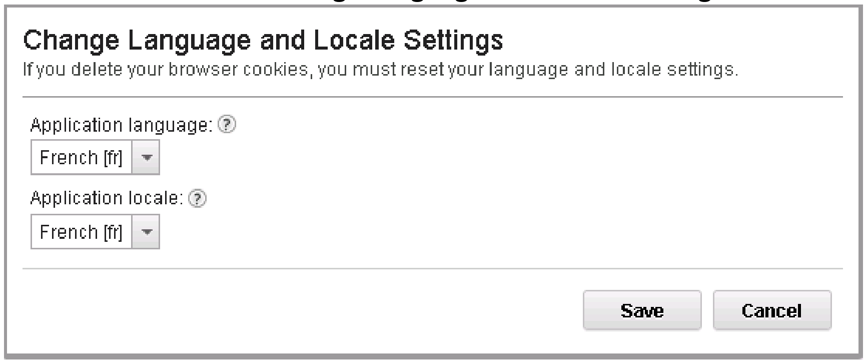
Enter **filenetadmin** for the **User name**.
Enter **IBMFileNetP8** for the **Password**.
Click **Log In**.



2. At the top of the menu bars, next to your login id (filenetadmin), click on the down arrow to see the user settings menu.



3. Click on the **Change Language and Locale Settings** menu item.



Change Language and Locale Settings

If you delete your browser cookies, you must reset your language and locale settings.

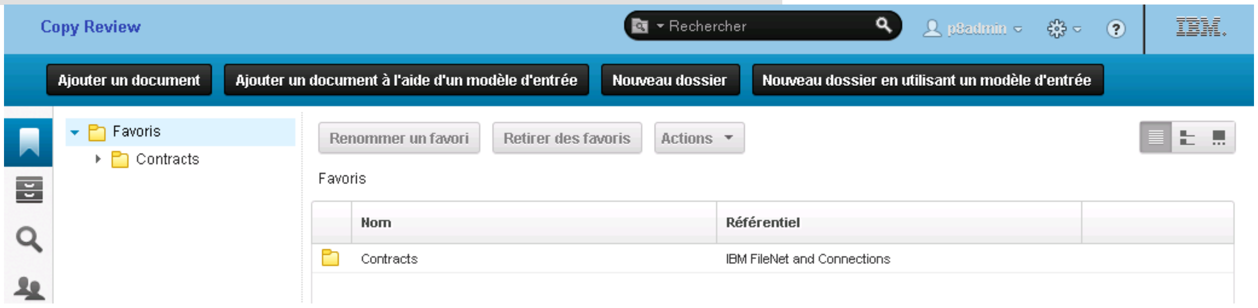
Application language: French [fr]

Application locale: French [fr]

4. Change your settings for both Application Language and Application Locale to **French**. Click **Save**.

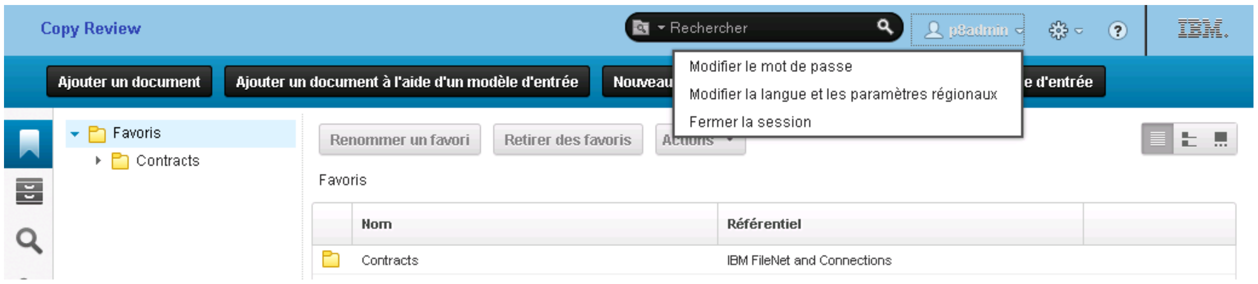
WHEN YOU CHOOSE BADLY

Some languages are read left to right, and down to up. If you can't read a language, and you choose it, you can always delete your browser cookies to get back to the default browser language setting.



Nom	Référentiel
Contracts	IBM FileNet and Connections

5. Now menu items and dialogs are displayed in French. Document names and metadata will retain whatever original language they were created or set in.

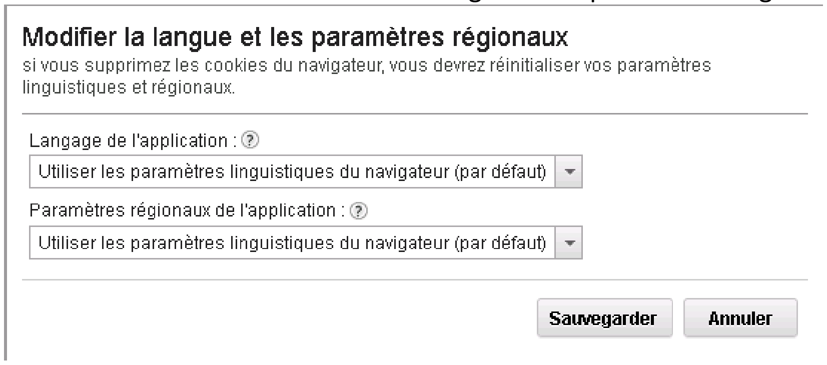


Favoris

Nom	Référentiel
Contracts	IBM FileNet and Connections

If you don't want to do the rest of the lab in French, or you prefer another language, perform the following two steps to reset to your preferred language.

6. Click the down arrow next to the login ID (filenetadmin) again to get the settings menu. This time it reads "Modifier la langue et les paramètres régionaux." – choose this item.



Modifier la langue et les paramètres régionaux

si vous supprimez les cookies du navigateur, vous devrez réinitialiser vos paramètres linguistiques et régionaux.

Langage de l'application : Utiliser les paramètres linguistiques du navigateur (par défaut)

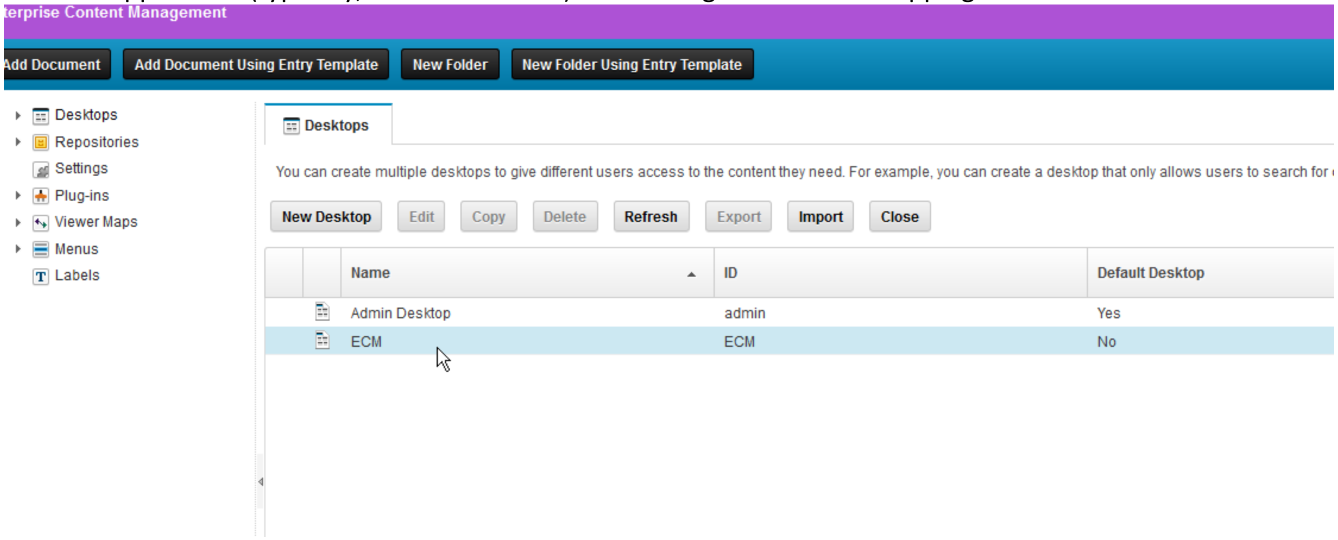
Paramètres régionaux de l'application : Utiliser les paramètres linguistiques du navigateur (par défaut)

7. For both Application Language and Application Locale, scroll up to the top of the list, to the one **ending in (par défault)**. Click **Sauvegarder**.

It is that simple for your international users to be supported in their language of preference.

B. Change the Viewer

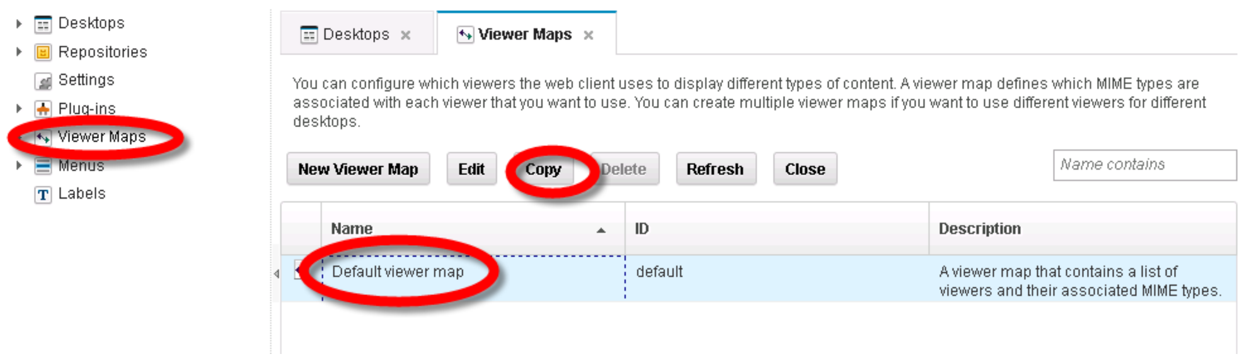
What if you want to compare and use the viewer for Word Documents instead of the default application (typically, Microsoft Office)? Just change the viewer mapping!



You can create multiple desktops to give different users access to the content they need. For example, you can create a desktop that only allows users to search for

Name	ID	Default Desktop
Admin Desktop	admin	Yes
ECM	ECM	No

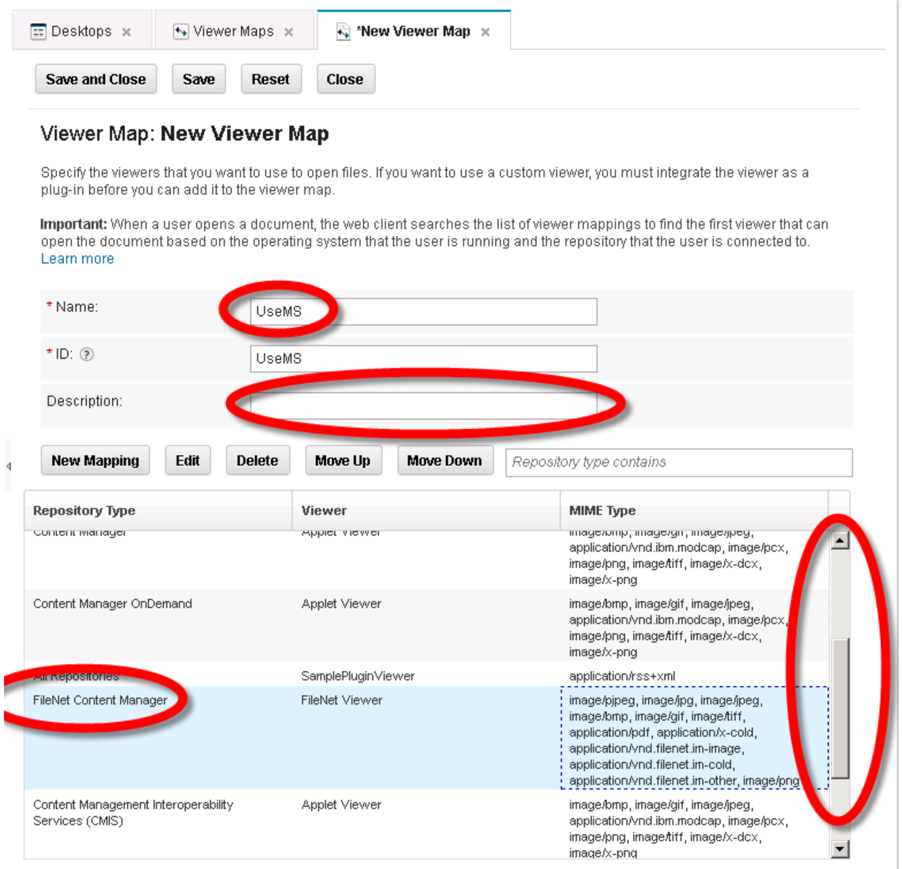
8. This time, instead of opening the Admin Desktop, we'll click on the **Open Administration View** button on the left.



You can configure which viewers the web client uses to display different types of content. A viewer map defines which MIME types are associated with each viewer that you want to use. You can create multiple viewer maps if you want to use different viewers for different desktops.

Name	ID	Description
Default viewer map	default	A viewer map that contains a list of viewers and their associated MIME types.

9. In the Admin view, go to the **Viewer Maps** tool by selecting it in the left pane. The Viewer Maps tab will appear on the right.
10. Click on the **Default Viewer map** and click **Copy**.
11. Name it **UseMS.** The ID will fill in by default – this can be altered but it is easier to maintain symbolic and display names that are the same. Clear the description field.



Viewer Map: **New Viewer Map**

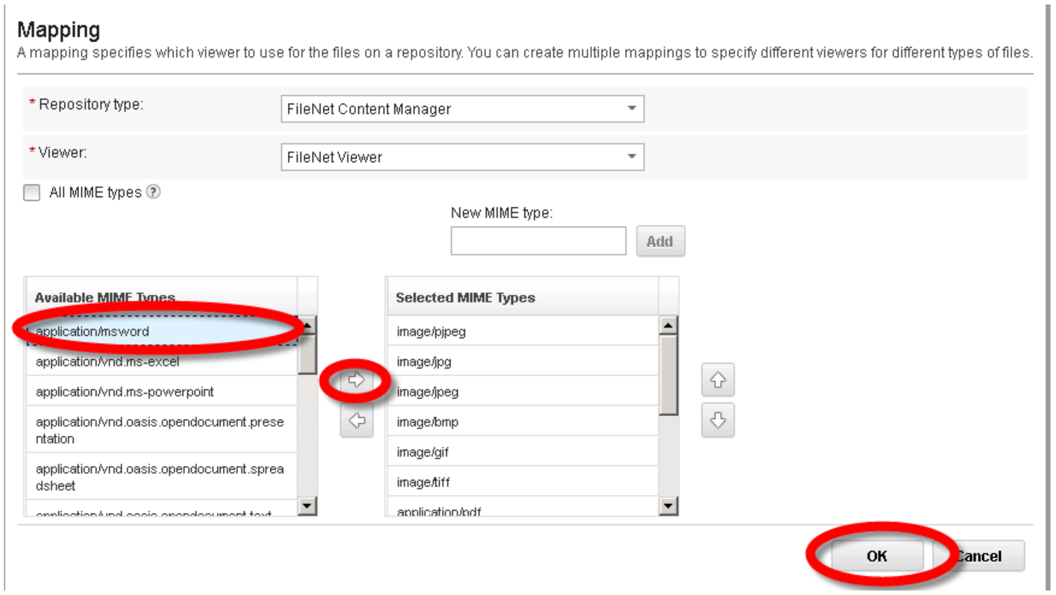
Specify the viewers that you want to use to open files. If you want to use a custom viewer, you must integrate the viewer as a plug-in before you can add it to the viewer map.

Important: When a user opens a document, the web client searches the list of viewer mappings to find the first viewer that can open the document based on the operating system that the user is running and the repository that the user is connected to.
Learn more

* Name: UseMS
* ID: UseMS
Description:

Repository Type	Viewer	MIME Type
Content Manager	Applet Viewer	image/bmp, image/gif, image/jpeg, application/vnd.ibm.modcap, image/pcx, image/png, image/tiff, image/x-dcx, image/x-png
Content Manager OnDemand	Applet Viewer	image/bmp, image/gif, image/jpeg, application/vnd.ibm.modcap, image/pcx, image/png, image/tiff, image/x-dcx, image/x-png
All Repositories	SamplePluginViewer	application/rss+xml
FileNet Content Manager	FileNet Viewer	image/pjpeg, image/jpg, image/jpeg, image/bmp, image/gif, image/tiff, application/pdf, application/x-cold, application/vnd.filenet.im-image, application/vnd.filenet.im-cold, application/vnd.filenet.im-other, image/png
Content Management Interoperability Services (CMIS)	Applet Viewer	image/bmp, image/gif, image/jpeg, application/vnd.ibm.modcap, image/pcx, image/png, image/tiff, image/x-dcx, image/x-png

12. In the table below, scroll down to click the row for the repository type **FileNet Content Manager** and viewer **FileNet Viewer**. Click **Edit** in the menu above. The Mapping dialog will appear.


Mapping

A mapping specifies which viewer to use for the files on a repository. You can create multiple mappings to specify different viewers for different types of files.

* Repository type: FileNet Content Manager

* Viewer: FileNet Viewer

All MIME types

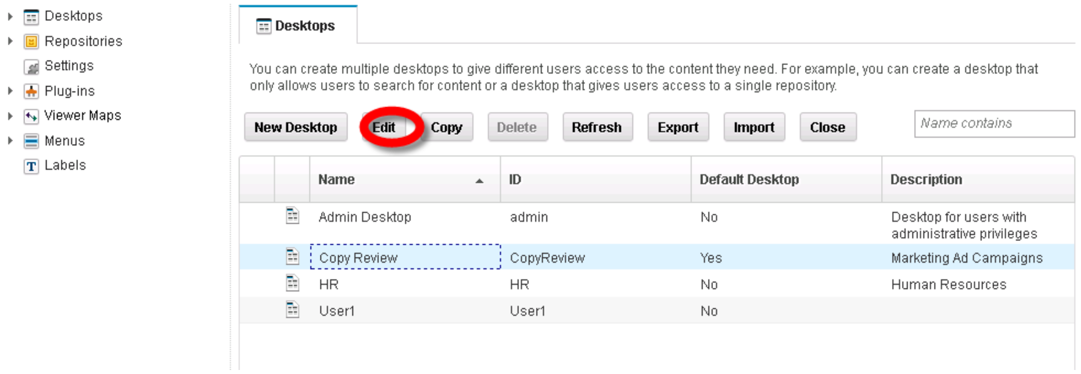
New MIME type:

Available MIME Types
application/msword
application/vnd.ms-excel
application/vnd.ms-powerpoint
application/vnd.oasis.opendocument.presentation
application/vnd.oasis.opendocument.spreadsheet

Selected MIME Types
image/pjpeg
image/jpg
image/jpeg
image/bmp
image/gif
image/tiff
application/pdf

13. In the **Available Mime Types** list on the left, select mime type **application/msword** and move it to the **Selected Mime Types** column using the right arrow button.

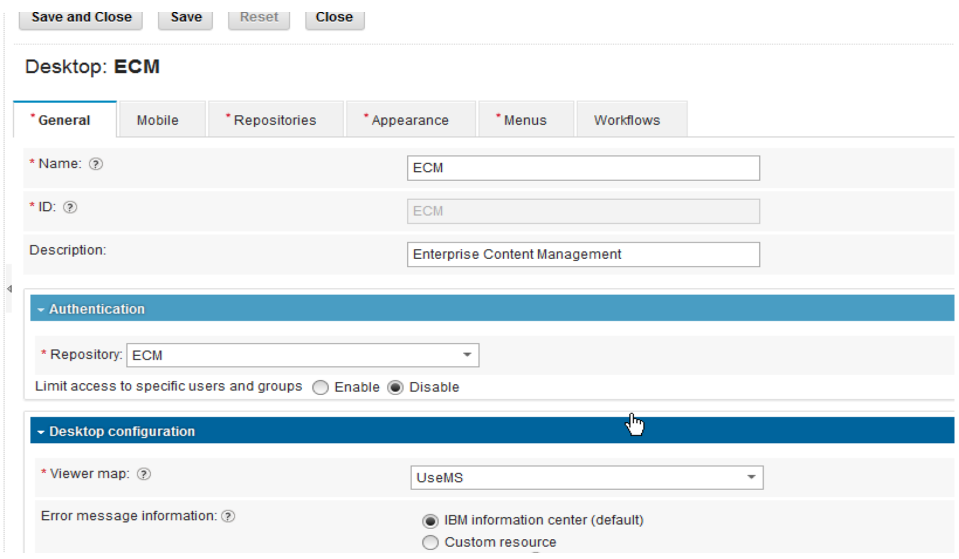
14. Click **OK**
15. Click the **Save and Close** button. The **UseMS** viewer map now appears on the list. Close the Viewer maps tab.



You can create multiple desktops to give different users access to the content they need. For example, you can create a desktop that only allows users to search for content or a desktop that gives users access to a single repository.

Name	ID	Default Desktop	Description
Admin Desktop	admin	No	Desktop for users with administrative privileges
Copy Review	CopyReview	Yes	Marketing Ad Campaigns
HR	HR	No	Human Resources
User1	User1	No	

16. On the **Desktops** tab, select the ECM desktop and click **Edit**. It will take a moment to connect and refresh properties.
17. On the **General** tab in the **Viewer Map** dropdown list, select **UseMS**.


Save and Close | Save | Reset | Close

Desktop: **ECM**

* General | Mobile | * Repositories | * Appearance | * Menus | Workflows

* Name: ECM

* ID: ECM

Description: Enterprise Content Management

Authentication

* Repository: ECM

Limit access to specific users and groups ( ) Enable (●) Disable

Desktop configuration

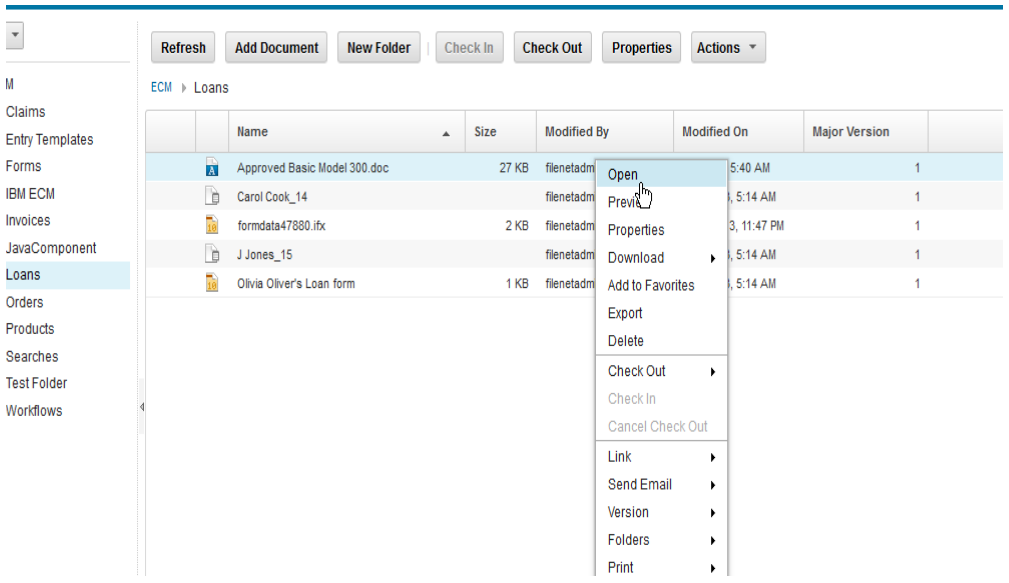
* Viewer map: UseMS

Error message information: (●) IBM information center (default) ( ) Custom resource

18. Click the **Save and Close** button.

19. Logout, close the browser, then reopen it and login to see the changes in the **Copy Review** desktop.

20. When you select a Word document (such as ECM**\Loans\Approved Basic Model 300.doc)** and double click on it, the viewer will open instead of Word.

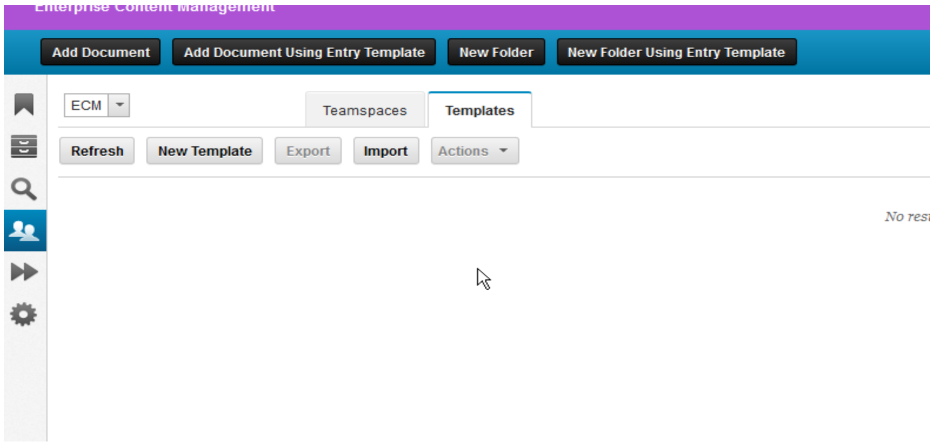


Name	Size	Modified By	Modified On	Major Version
Approved Basic Model 300.doc	27 KB	filenetadm	5:40 AM	1
Carol Cook_14		filenetadm	, 5:14 AM	1
formdata47880.ifx	2 KB	filenetadm	3, 11:47 PM	1
J Jones_15		filenetadm	, 5:14 AM	1
Olivia Oliver's Loan form	1 KB	filenetadm	, 5:14 AM	1

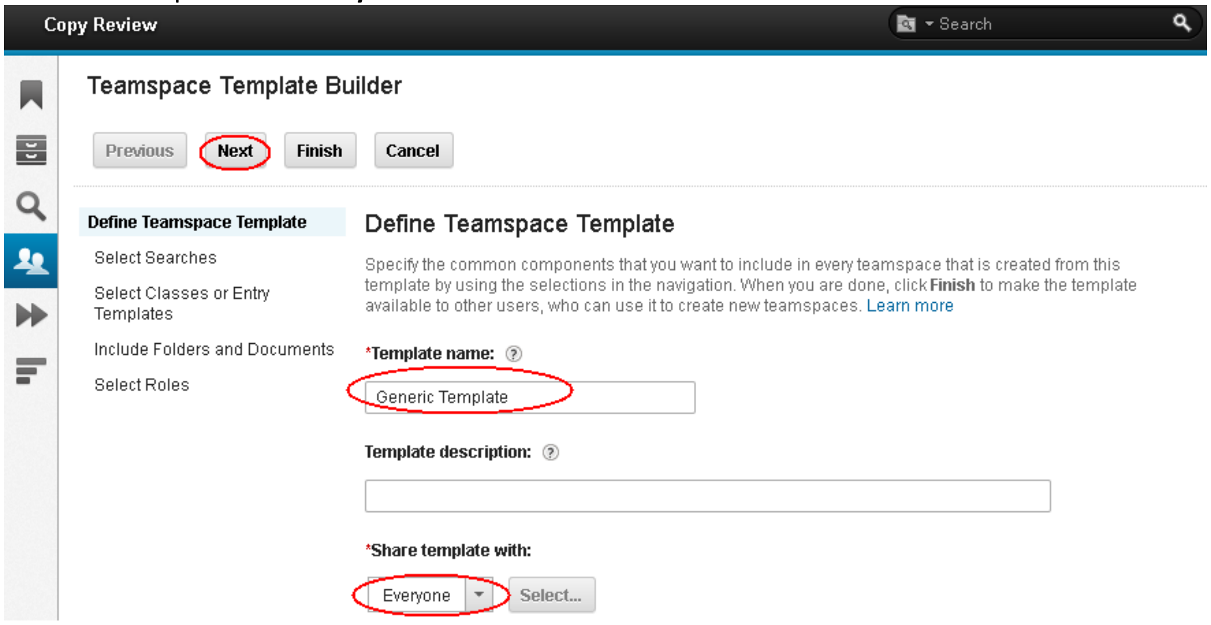
3. Creating a Teamspace Template

Another way that teams can work together on documents is Teamspaces. These spaces combine searches, browsing, documents, and people to make working on content efficient. This lab section walks through this functionality. Templates are a way to help teams create their Teamspaces more consistently and efficiently by giving a customizable blueprint of which folders, documents, entry templates, and searches match their requirements.

21. Choose the **Teamspace** icon from the feature pane on the left and Click on the **Templates** tab.


Enterprise Content Management
Add Document
Add Document Using Entry Template
New Folder
New Folder Using Entry Template
ECM
Teamspaces
Templates
Refresh
New Template
Export
Import
Actions
No resu

22. Create a new template by clicking on **New Template**, called **Generic Template**. Share the template with **Everyone**. Click **Next**.


Copy Review
Search
Teamspace Template Builder
Previous
Next
Finish
Cancel
Define Teamspace Template
Select Searches
Select Classes or Entry Templates
Include Folders and Documents
Select Roles
Define Teamspace Template

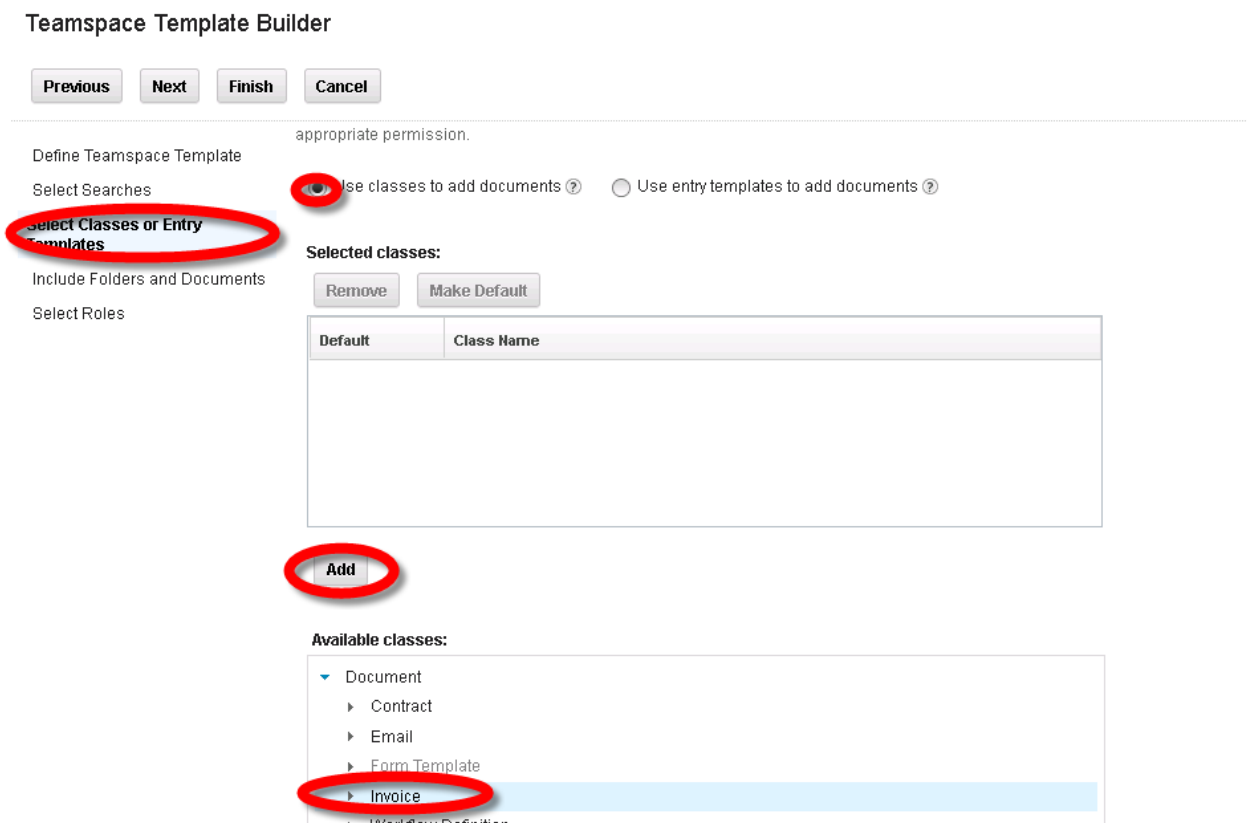
Specify the common components that you want to include in every teamspace that is created from this template by using the selections in the navigation. When you are done, click **Finish** to make the template available to other users, who can use it to create new teamspaces. Learn more

*Template name: Generic Template

Template description:

*Share template with: Everyone

23. Click **Next** again (or optionally create and/or add a search if you desire).


Teamspace Template Builder
Previous
Next
Finish
Cancel
Define Teamspace Template
Select Searches
Select Classes or Entry Templates
Include Folders and Documents
Select Roles
Remove
Make Default
Add

appropriate permission.

Use classes to add documents ? ○ Use entry templates to add documents ?

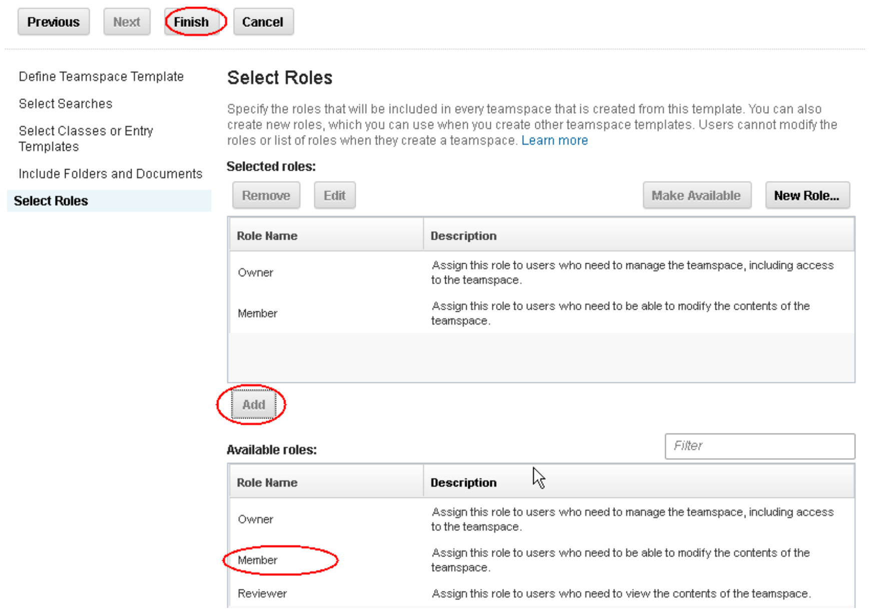
Selected classes:

Default	Class Name

Available classes:

- Document
  - Contract
  - Email
  - Form Template
  - Invoice

24. Select **Use Classes to Add Documents**. A blank box for Selected Classes appears, and in the bottom, a set of Available Classes appears.
25. Select the class **Invoice** and click **Add**. Click **Next**.
26. Click **Next** again (or optionally add folders and documents if you desire).



Select Roles

Specify the roles that will be included in every teamspace that is created from this template. You can also create new roles, which you can use when you create other teamspace templates. Users cannot modify the roles or list of roles when they create a teamspace. Learn more

Selected roles:

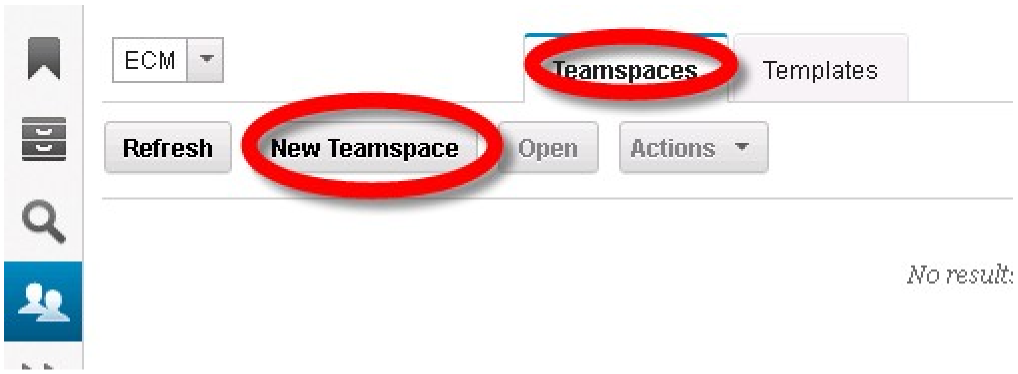
Role Name	Description
Owner	Assign this role to users who need to manage the teamspace, including access to the teamspace.
Member	Assign this role to users who need to be able to modify the contents of the teamspace.

Available roles:

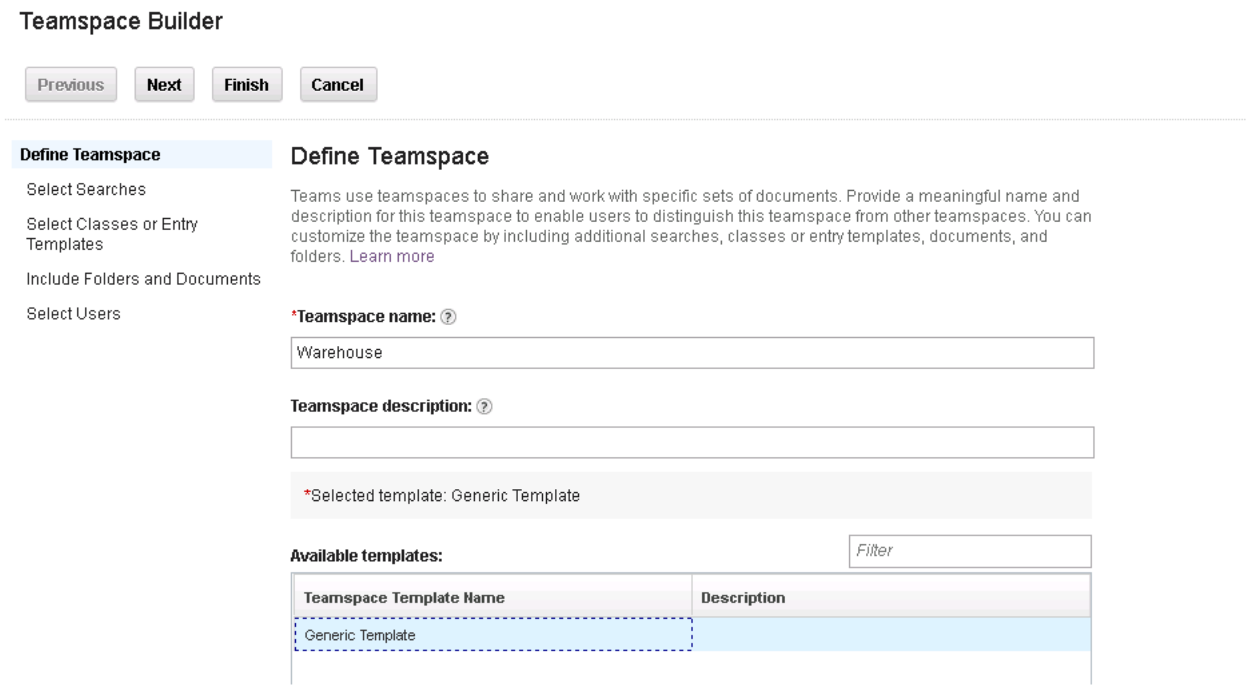
Role Name	Description
Owner	Assign this role to users who need to manage the teamspace, including access to the teamspace.
Member	Assign this role to users who need to be able to modify the contents of the teamspace.
Reviewer	Assign this role to users who need to view the contents of the teamspace.

27. Add the **Member** role by selecting it and click add. Click **Finish**.

Now the template is ready to use. Remember, it's a blueprint – the user creating the teamspace can alter it to fit their needs.



28. Create a new Teamspace there by clicking on the **Teamspace** tab. Click on **New Teamspace.**
29. Name it **Warehouse**. Use the **Generic Template** you just created by clicking on it. Click **Next**.



Define Teamspace

Teams use teamspaces to share and work with specific sets of documents. Provide a meaningful name and description for this teamspace to enable users to distinguish this teamspace from other teamspaces. You can customize the teamspace by including additional searches, classes or entry templates, documents, and folders. Learn more

*Teamspace name: Warehouse

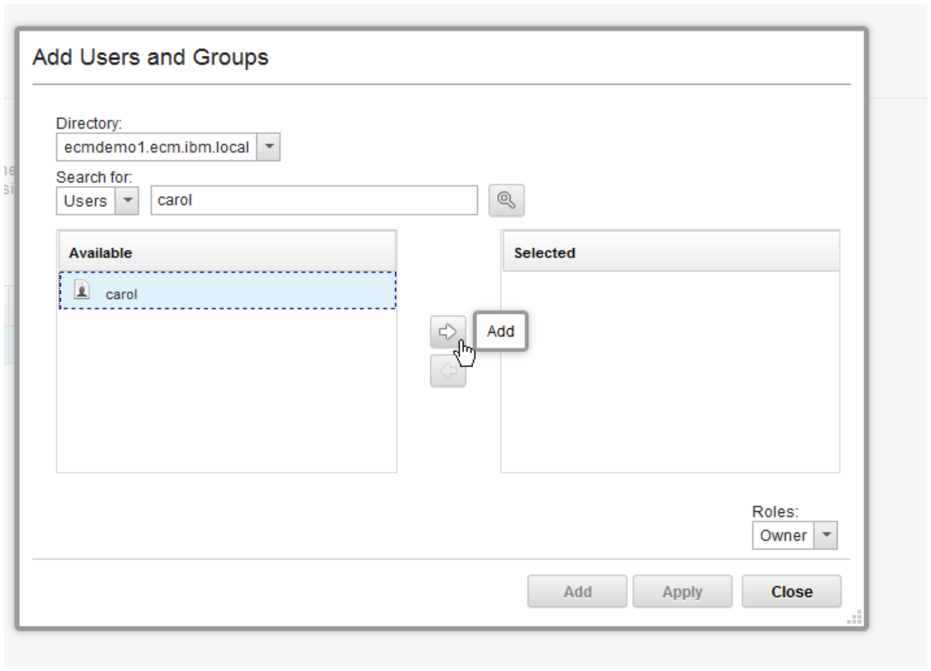
Teamspace description:

*Selected template: Generic Template

Available templates:

Teamspace Template Name	Description
Generic Template	

30. Click **Next** again (or create and add a search if you desire).
31. You will see the class **Invoice** that you added in the template. Click **Next**.
32. Click **Next** again (or add folders and documents if you desire).
33. filenetadmin is already an owner. Add the user **carol** by clicking on **Add Users and Groups**


Add Users and Groups

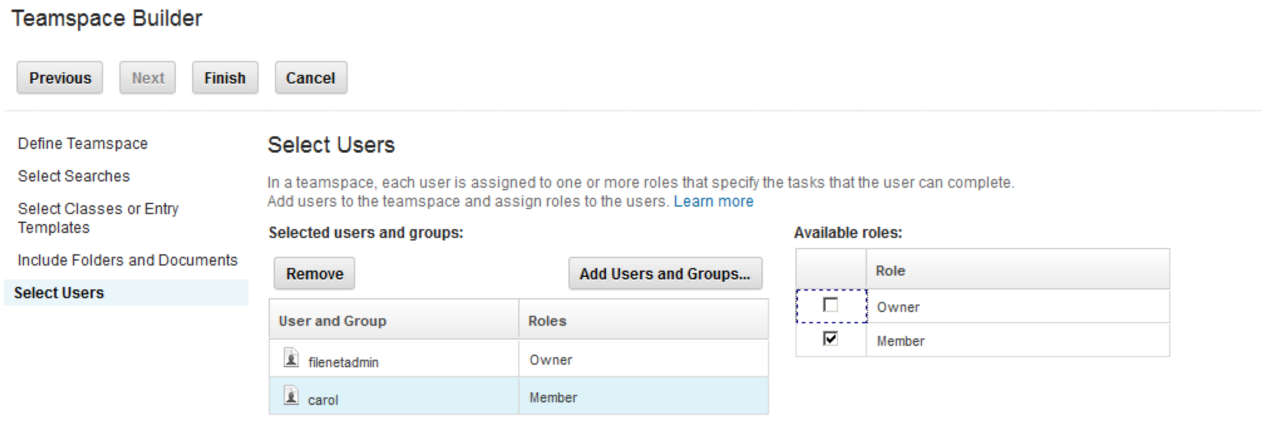
Directory: ecmdemo1.ecm.ibm.local

Search for: Users carol

Available	Selected
carol	

Roles: Owner

34. Enter the letter **c** in the Users box, and click the **search button**.
35. Select **carol**, and click on the arrow to add him. Then click the **Add** button to add him as a member of the team.


Teamspace Builder

Select Users

In a teamspace, each user is assigned to one or more roles that specify the tasks that the user can complete. Add users to the teamspace and assign roles to the users. Learn more

Selected users and groups:

User and Group	Roles
filenetadmin	Owner
carol	Member

Available roles:

	Role
☐	Owner
☑	Member

36. Select the user carol in the list, and then select a role by checking the box next to it, or remove him from a role by unchecking it.
37. Click **Finish**.

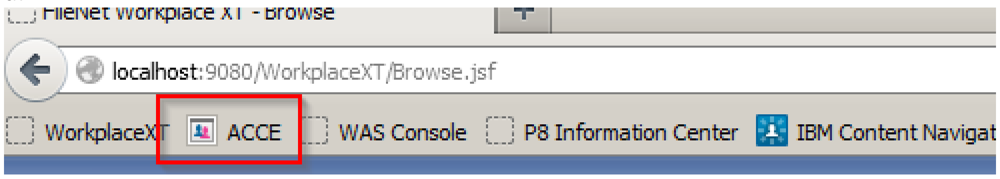
Now your Teamspace is ready for collaboration.

38. Finishing your Teamspace navigates you right into it. You can customize it further by changing the layout. Drag some widgets around by clicking and dragging on the title of the widget, such as **Search** and **Browse**.

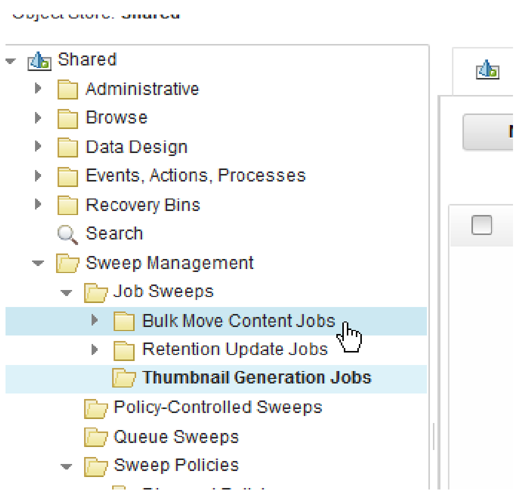
Teamspaces can be customized by users as well as administrators. It is easy to quickly set up a place to collaborate on documents with coworkers.

4. Thumbnails

Thumbnail images can be created for new documents and for existing documents. Images for existing documents can be generated automatically during a sweep of all objects by specifying the class of the objects that thumbnail images should be generated for. ACCE link is provided as a short in the browser toolbar

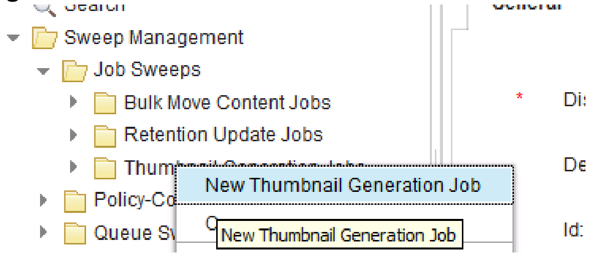

FileNet Workplace XT - Browse
localhost:9080/WorkplaceXT/Browse.jsf
WorkplaceXT
ACCE
WAS Console
P8 Information Center
IBM Content Navigat

39. In ACCE navigate to Shared **> Sweep Management > Job Sweeps > Thumbnail Generation Jobs**



40.

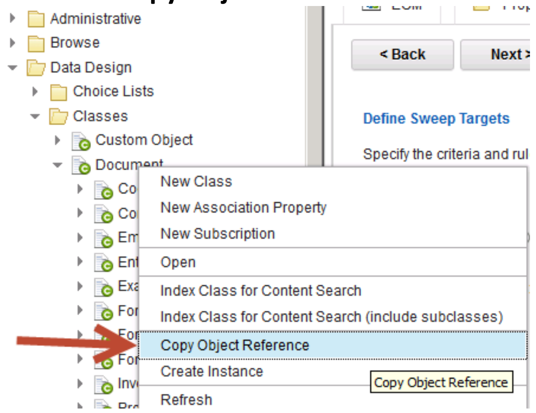
Right click on **Thumbnail Generation Job > New Thumbnail Generation Job**



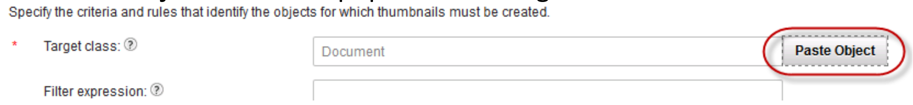
Provide the name **Thumbnail Generation**

Click **Next**

This section we need to provide the Target class. Locate the Document root class, right-click and select **Copy Object Reference**



Click **Paste Object** which will populate the Target Class field


Specify the criteria and rules that identify the objects for which thumbnails must be created.

* Target class: Document

Filter expression:

Select **Include Subclasses**

Click **Next**, and click OK on the window that opens

Leave the dates blank. Click **Next**

Click **Finish**

Click **Open**

Select **Enabled**, click **Save**

Click **Close**

41. You have now completed and created the thumbnail sweep job. The final step is ensuring that the Thumbnail system property is available in the repository.

Access the ICN Admin Desktop

Open the ECM repository

Click **Connect**

In the **System Properties** tab add the **Thumbnails*** property to Selected Properties (note it is already added)

In the **Browse** tab add the **Thumbnails*** property to Selected Properties

In the **Search** tab add the **Thumbnails*** property to Selected Properties

Click **Save and Close**

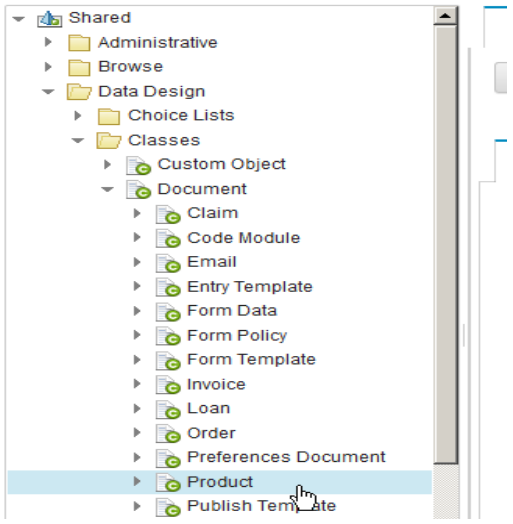
42. Thumbnails are now enabled. It will take a little bit of time to create the thumbnails. Congratulations!

Retention and Recovery Bin

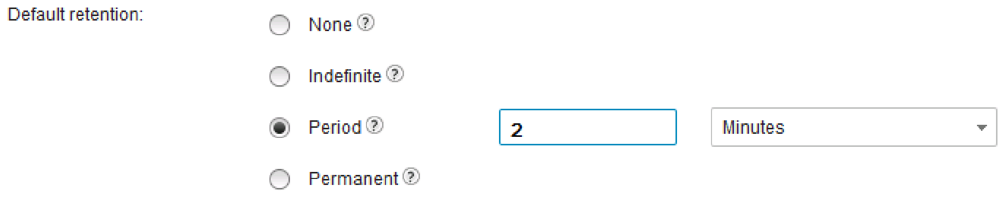
Expiration Manager

Retention specifies a fixed amount of time that an object must be kept or retained. You can set retention for objects directly or the object can default to the retention that is specified for the object class. The retention that you specify for an object can later be applied to the object content after it is checked into a fixed storage area.
The retention period of a large number of objects can be automatically updated during a sweep of all objects, by using the class of the object or the state of the object's properties. You can specify a new retention date directly, or you can cause the new date to be calculated.

1. Access ACCE
2. Navigate to Shared **> Data Design > Classes > Document** expand Product



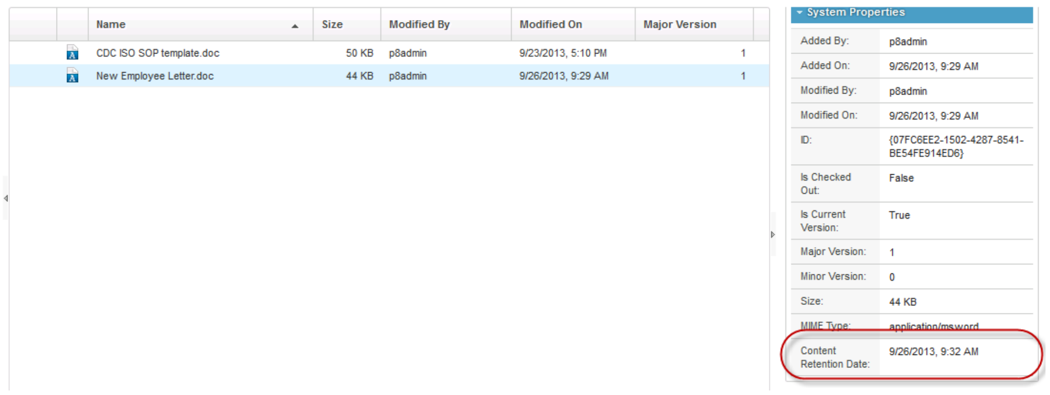
3. Navigate to the **Retention** tab
4. Hover over the different ? icons to discover what each retention period means. Select the **Period** option and set it to **2 minutes**


Default retention:
None
Indefinite
Period ● 2 Minutes
Permanent

5. **Save** and **Close**
6. The Content Retention Date system property needs to be made available in the repository.
7. Access the ICN Admin Desktop
8. Open the ECM repository

9. Click **Connect**
10. In the **System Properties** tab add the **Content Retention Date** property to Selected Properties
11. **Save and Close**
12. Add a new document to the **Product** document class

13. Select the document and view the system properties. There is the retention date and time

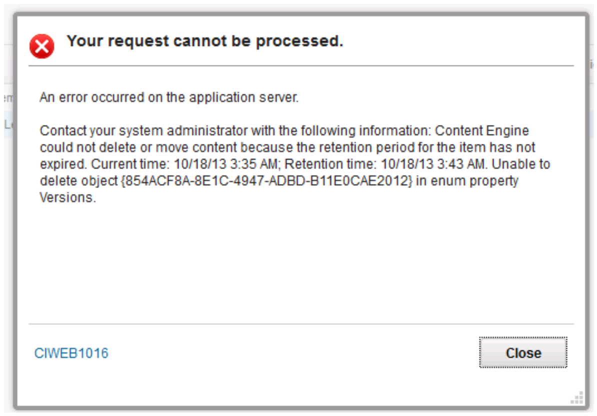


Name	Size	Modified By	Modified On	Major Version
CDC ISO SOP template.doc	50 KB	p8admin	9/23/2013, 5:10 PM	1
New Employee Letter.doc	44 KB	p8admin	9/26/2013, 9:29 AM	1

System Properties

Added By:	p8admin
Added On:	9/26/2013, 9:29 AM
Modified By:	p8admin
Modified On:	9/26/2013, 9:29 AM
ID:	{07FC6EE2-1502-4287-8541-BE54FE914ED6}
Is Checked Out:	False
Is Current Version:	True
Major Version:	1
Minor Version:	0
Size:	44 KB
MIME Type:	application/msword
Content Retention Date:	9/26/2013, 9:32 AM

14. Try to delete it, and you will receive the following message. P8 does not allow you delete the document until the retention period has ended. By the time we finish creating the sweep job we will be able to delete the document.



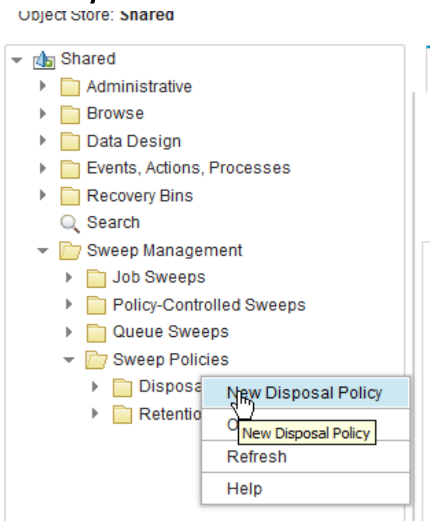
Your request cannot be processed.

An error occurred on the application server.

Contact your system administrator with the following information: Content Engine could not delete or move content because the retention period for the item has not expired. Current time: 10/18/13 3:35 AM; Retention time: 10/18/13 3:43 AM. Unable to delete object {854ACF8A-8E1C-4947-ADBD-B11E0CAE2012} in enum property Versions.

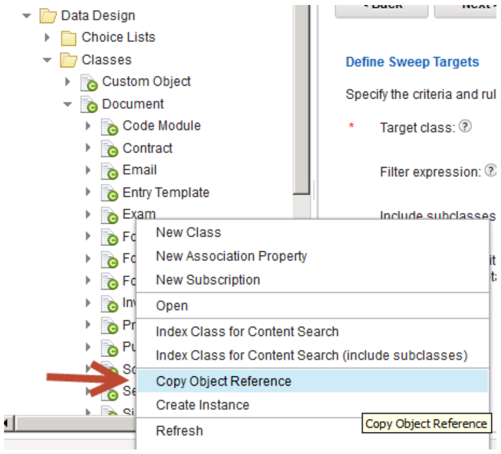
CIWEB1016

15. Lets create a sweep job to delete it.
16. In ACCE navigate to Shared **> Sweep Management > Disposal Retention** right-click **New Disposal Policy**

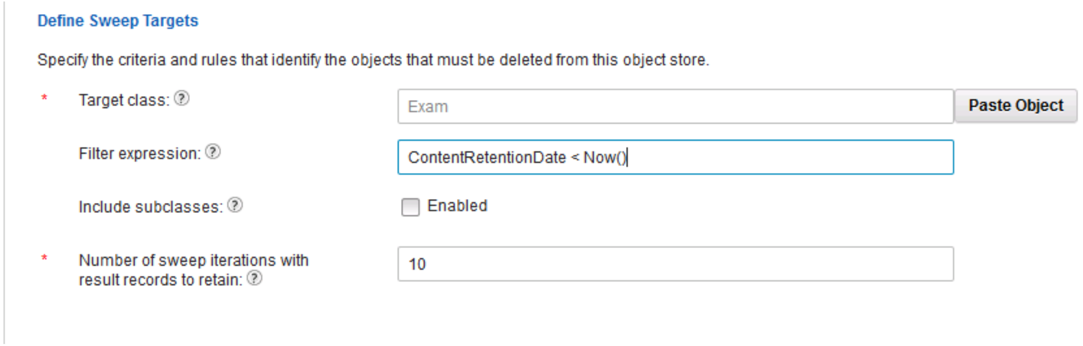


17. Set the **Display Name** to **Product Disposal Sweep**
18. Scroll down and check **Enable Disposal Sweep**
19. Click **Next**

20. Navigate to Shared **> Data Design > Classes > Document > Product** right-click **Copy Object Reference**



21. Click **Paste Object** which will populate the Target Class field
22. Set Filter = **ContentRetentionDate < Now()**



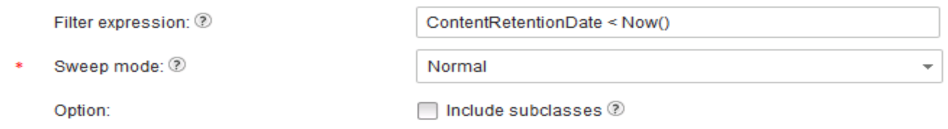
Define Sweep Targets

Specify the criteria and rules that identify the objects that must be deleted from this object store.

* Target class:	Exam
Filter expression:	ContentRetentionDate < Now()
Include subclasses:	☐ Enabled
* Number of sweep iterations with result records to retain:	10

Note - In our lab we are using the Content Retention Date, however you could use the Creation Date, Date Content Last Accessed, or a custom property (Contract=Closed and Date Last Modified > 5 Years Ago).

23. Click **Next**
24. Review and Click **Finish**
25. Click **Open**
26. Check the **Enable** check box
27. Update the Sweep Mode to **Normal** and **Number of sweep iterations with result records to retain** to **0**



Filter expression:	ContentRetentionDate < Now()
* Sweep mode:	Normal
Option:	☐ Include subclasses

Note - if you set Sweep Mode to Preview it will run the sweep job and add the results to the Sweep Results Tab. This is good for testing to see if your sweep works without affecting your documents.

28. Click **Save**
29. Click **Refresh**
30. You will note the Completed Iterations = 1 and that the Examined Objects = 1
31. Go back to ICN and note that your document is gone.

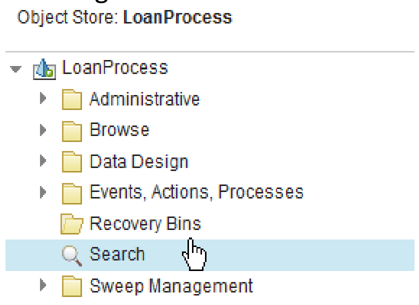
Congratulations you have expired a document.

Recovery Bin

A *recovery bin* is a container that you can move objects to when you want to delete them. When objects are moved to a recovery bin instead of being permanently deleted, the objects are marked for deletion. Marking objects for deletion makes it possible to recover the objects later. If an object that has a cascading relationship with other objects is marked for deletion, those objects are also marked for deletion. Multiple recovery bins can be created in each object store, and objects can be permanently deleted by emptying the recovery bins that the objects are in.

Today this is only exposed from a UI standpoint through IBM Connections (Connections Enterprise Content Edition)

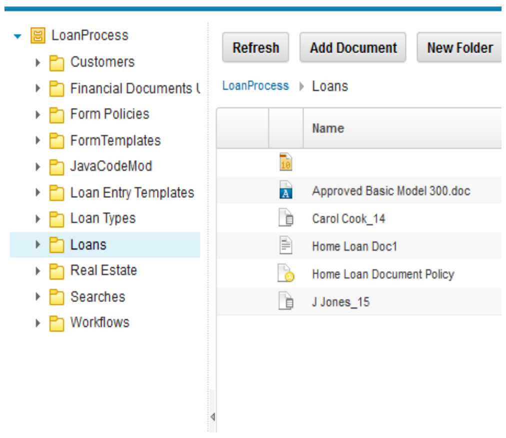
32. Access ACCE
33. Navigate to **LoanProcess > Recovery Bin** right-click **New Recovery Bin**



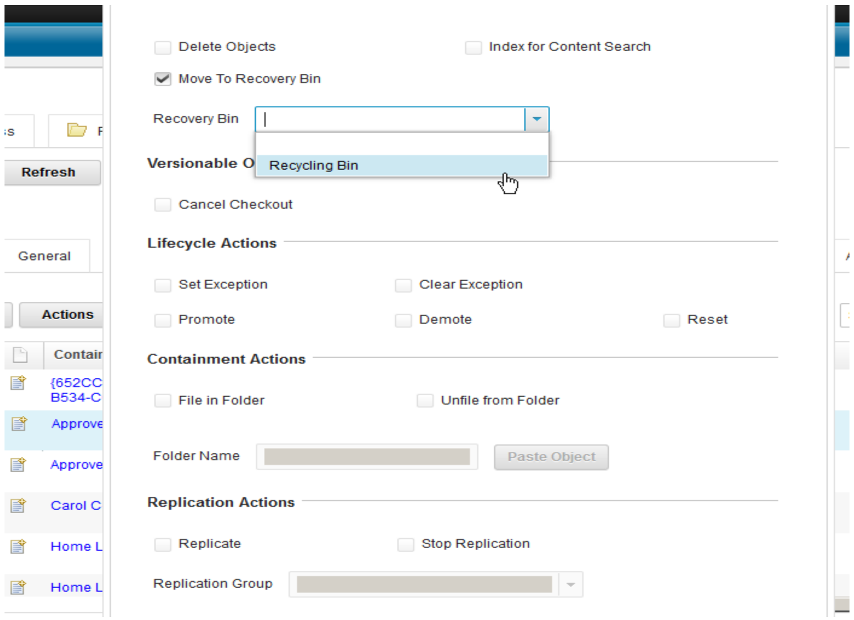
34. Set the display name to **Recycling Bin**
35. Click **Next**
36. Click **Finish**
37. Click **Open**
38. You will note that it is empty. How about we put something into it.

__1. Go to ICN – http://localhost:9080/navigator/?desktop=LoanProcess

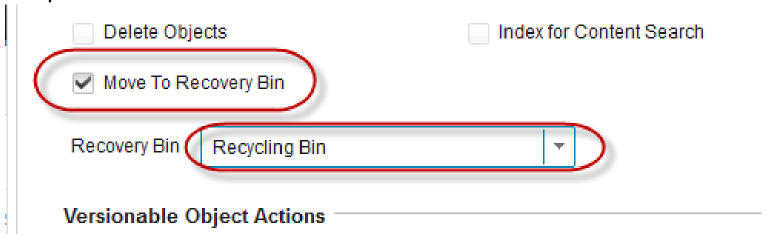
__2. Add a document into the Loans folder in the Loan Process object store. Use the default Document class and give the Document Title as Recycled Document



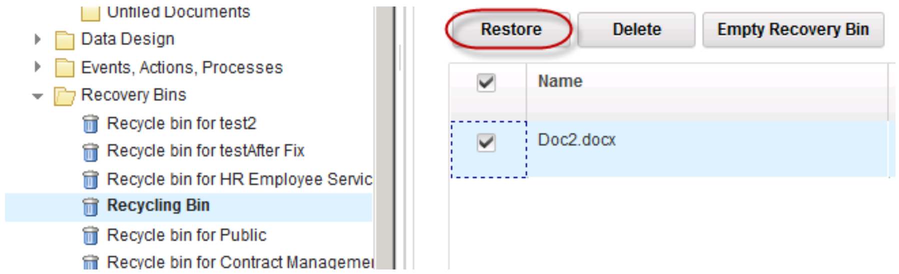
__3. Go back to **ACCE**
__4. Locate your document **Browse > Root Folder > Loans**
__5. Select your document and click **Action > Batch Operations**



__6. In the Batch Operations window select **Move To Recovery Bin** and select the **Recycling Bin** from the drop-down



__7. Click **Ok**
__8. Navigate back to ICN (LoanProcess desktop) and refresh Loans folder view. You will note that you document is gone.
__9. Navigate back to ACCE
__10. Access **Recycling Bin** and you will see your document is there
__11. Select the document and click **Restore**



__12. Your Recovery Bin is now empty
__13. Navigate back to ICN (LoanProcess desktop) and refresh Loans folder view. You will note that you document is back.
__14. If you like you can repeat steps 8-10, but this time Delete the document from the Recovery Bin. Your document will be permanently purged from FileNet.

Lab Activity 2 – Customizing the Login Page

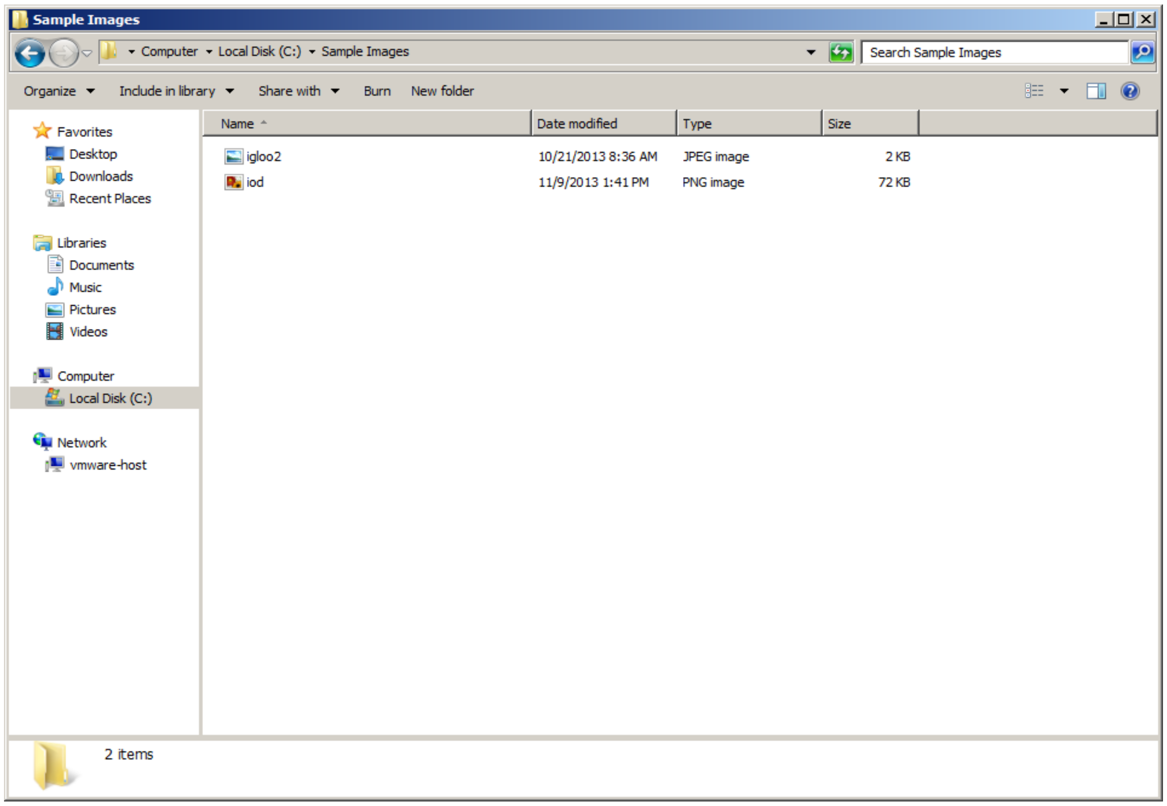
In the previous chapter you walked through the basic Content Navigator administration and familiarized yourself with all of the administration features. You also walked through a small exercise to create a simple Desktop.
In this chapter, we'll now take a look at how you can customize the desktop even further by changing the colors, logos, and URL's that the user will see when working with Content Navigator without having to write any code whatsoever.
Before we start with the Content Navigator administration steps, we'll first take a look at some sample image files and HTML files we'll use in this exercise.

Step 1 – Open Windows Explorer

Open Windows Explorer by clicking on right most icon in the Quick launch bar.

Step 2 – Open the Sample Images directory

Double click on the C:\Sample Images directory. Note the iod.png and igloo2.jpg files that are there. We'll be using this file to customize the login page logo later in the Navigator administration.



Name	Date modified	Type	Size
igloo2	10/21/2013 8:36 AM	JPEG image	2 KB
iod	11/9/2013 1:41 PM	PNG image	72 KB

Step 3 – Copy an image to the navigator custom directory

In this step we'll copy the image over to a directory in a web application so that the image file will be URL addressable.

Right click on the file and select Copy.

On the left hand browse tree, expand the directory tree to C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\navigator.ear\navigator.war\custom

Right click in the examples directory and choose Paste to add the iod.png and igloo2.jpg files to the directory.

Step 4 – Copy an html file to the WorkplaceXT images directory

This HTML file will be used to create a custom login help area on the ECM Desktop's login page.

In Windows Explorer, select the C:\Sample HTML Files directory.

Right click on the Login.html file and choose Copy.

On the left hand browse tree, expand the directory tree to C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\navigator.ear\navigator.war\custom

Right click in the examples directory and choose Paste to add the Login.html file to the directory.

When you're done your examples directory should look like the following screen.

Step 5 – Open the Content Navigator Administration

Double-click on the Content Navigator Administration icon on the desktop. Login using the filenetadmin/IBMFileNetP8 user and password.

Step 6 – Open the ECM Desktop

After logging in to administration you should automatically be placed on the Desktops panel. Double click on the ECM Desktop.

Step 7 – Select the Appearance Tab

Click on the Appearance tab.

Step 8 – Update the Appearance Tab

In this step we'll be making changes to the desktop's Banner and Login Page settings.

Next to the Login Page Logo label, change the radio button to URL.

In the URL text box, enter http://Base-Win2k8x64:9080/navigator/custom/ecm.jpg

Scroll down a bit and change the radio button next to Banner logo label to URL.

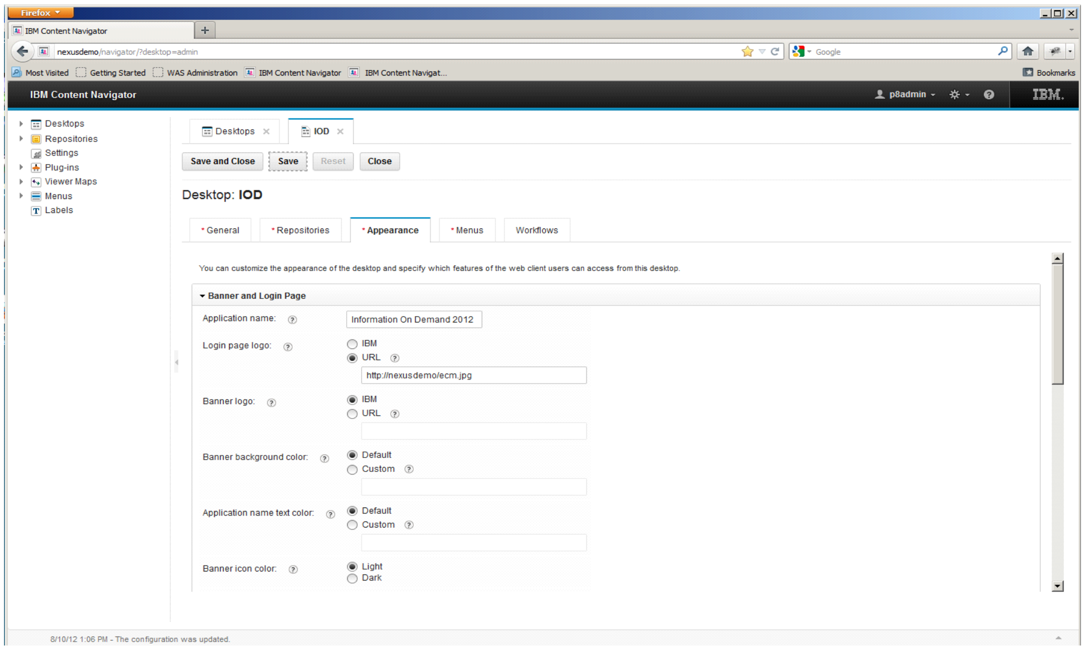
In the URL text box, enter http://Base-Win2k8x64:9080/navigator/custom/igloo2.jpg

Scroll down a bit and change the radio button next to Login page content to URL.

In the URL text box, enter http://Base-Win2k8x64:9080/navigator/custom/Login.html

Scroll down a bit and change the radio button next to Banner background color to Custom

In the text box enter the color: #8E3F3F



IBM Content Navigator

Desktop: IOD

You can customize the appearance of the desktop and specify which features of the web client users can access from this desktop.

Banner and Login Page

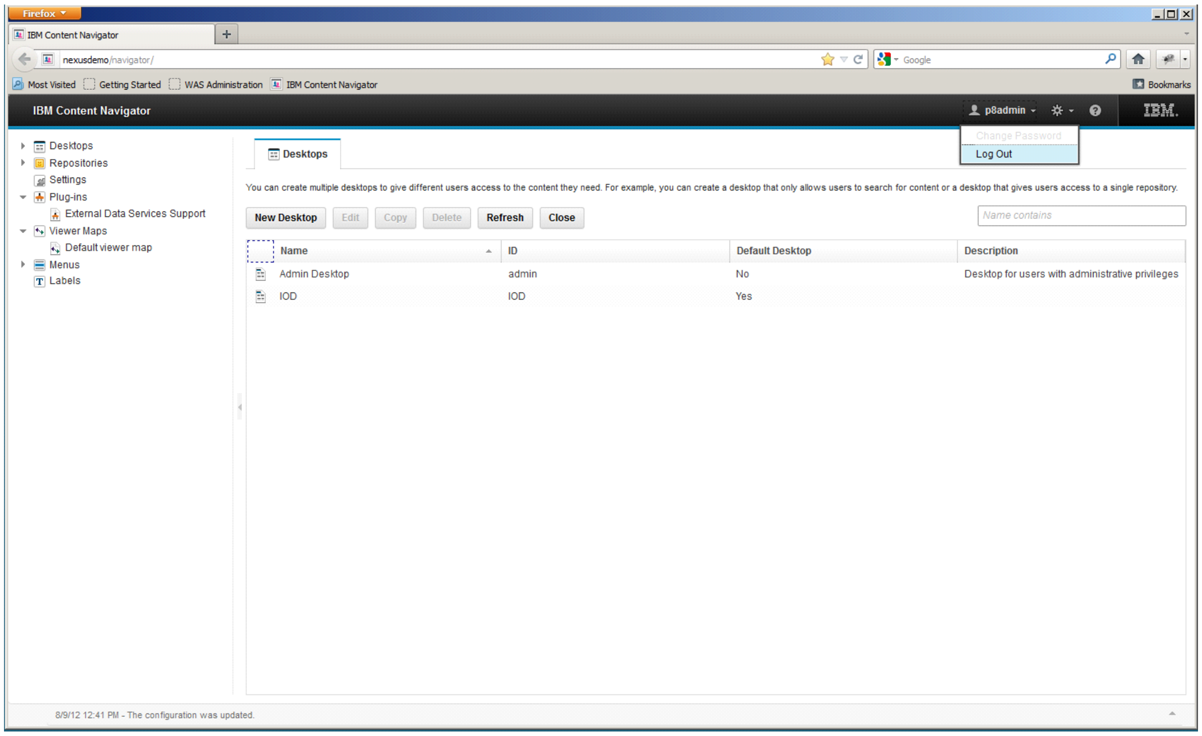
Application name:	Information On Demand 2012
Login page logo:	IBM [✓ URL] http://nexusdemo/ecm.jpg
Banner logo:	[✓ IBM] URL
Banner background color:	[✓ Default] Custom
Application name text color:	[✓ Default] Custom
Banner icon color:	[✓ Light] Dark

8/10/12 1:06 PM - The configuration was updated.

Click Save and Close.

Step 9 – Logout and Reopen the Browser

Click the down arrow next to the username in the upper right hand corner of the application and select **Log Out**.



IBM Content Navigator

Desktops

You can create multiple desktops to give different users access to the content they need. For example, you can create a desktop that only allows users to search for content or a desktop that gives users access to a single repository.

Name	ID	Default Desktop	Description
Admin Desktop	admin	No	Desktop for users with administrative privileges
IOD	IOD	Yes	

8/9/12 12:41 PM - The configuration was updated.

Click the **Log Out** button when prompted.

Close Internet Explorer.

Step 10 – Reload Content Navigator

Reopen Internet Explorer by clicking on the Internet Explorer icon in the Quick launch bar.

Enter http://Base-Win2k8x64:9080/navigator?desktop=ECM and hit Enter.

Note that your login page now has a section on the left that allows the user to get help with logging in to the application. Also note that the application now has a new icon in the bottom right corner of the login area.

Lab Activity 4 – Custom Labels

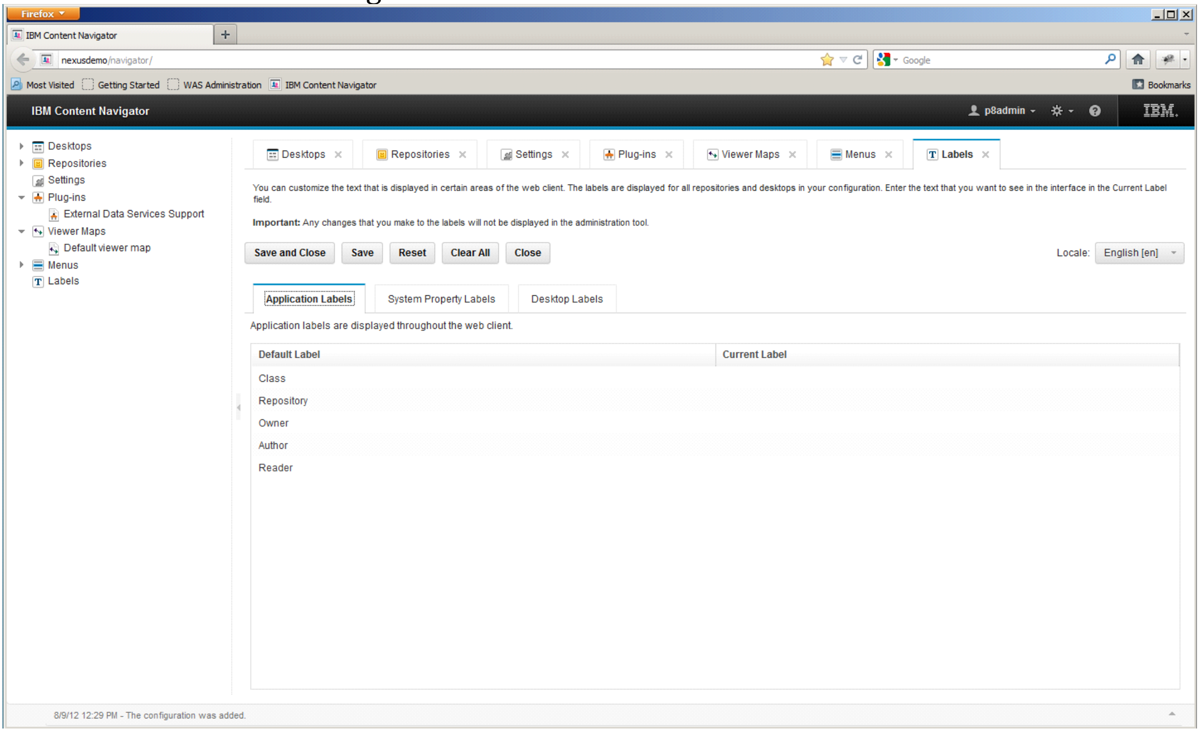
In this next Activity, we'll walkthrough how easily you can change the default Application labels used throughout Content Navigator. Again, this can be done without any development effort. Sensing a trend yet? The use-case here is that a customer that only has FileNet Content Manager may want to use the labels of Object class and Object store instead of Class and Repository. This activity will walk you through how to make this change and test it in the application.

Step 1 – Open the Content Navigator Administration

Double-click on the Content Navigator Administration icon on the desktop and login using **filenetadmin** as the user and **IBMFileNetP8** for the password.

Step 2 – Select the Labels Panel

On the left hand side of the screen, click on the **Labels** option in the tree. You should see a screen that looks like the following.



You can customize the text that is displayed in certain areas of the web client. The labels are displayed for all repositories and desktops in your configuration. Enter the text that you want to see in the interface in the Current Label field.

Important: Any changes that you make to the labels will not be displayed in the administration tool.

Application labels are displayed throughout the web client.

Default Label	Current Label
Class	
Repository	
Owner	
Author	
Reader	

8/9/12 12:29 PM - The configuration was added.

Step 3 – Edit the Application Labels

Here we'll actually update the **Application** labels called **Class** and **Repository** to new values.

Find the **Class label** in the **Default Label** column. Click in the space next to that label in the **Current Label** column.

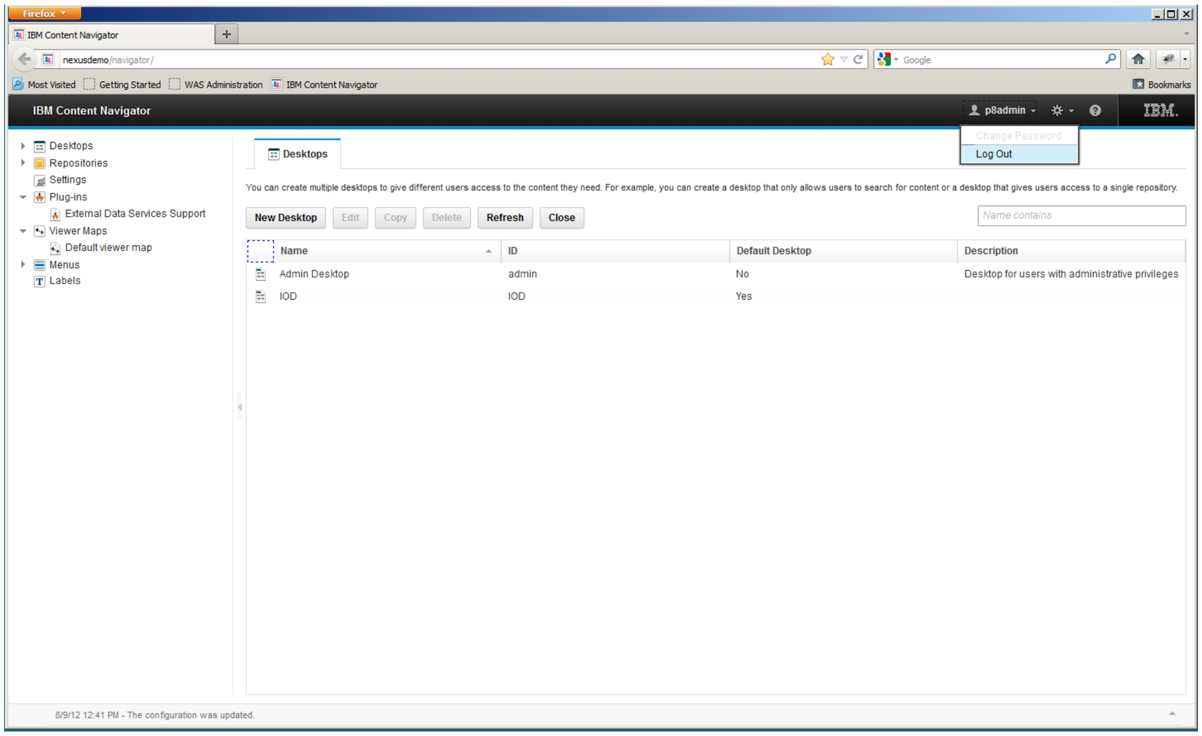
Enter **Object Class** as the new label.

Repeat steps 1 and 2 for **Repository**, this time using a new label of **Object Store**.

Click the **Save and Close** button.

Step 4 – Logout and Reopen the Browser

Click the down arrow next to the username in the upper right hand corner of the application and select **Log Out**.



You can create multiple desktops to give different users access to the content they need. For example, you can create a desktop that only allows users to search for content or a desktop that gives users access to a single repository.

Name	ID	Default Desktop	Description
Admin Desktop	admin	No	Desktop for users with administrative privileges
IOD	IOD	Yes	

8/9/12 12:41 PM - The configuration was updated.

Click the **Log Out** button when prompted.

Close IE.

Step 5 – Reload Content Navigator

Reopen IE by clicking on the IE icon in the Quick launch bar.

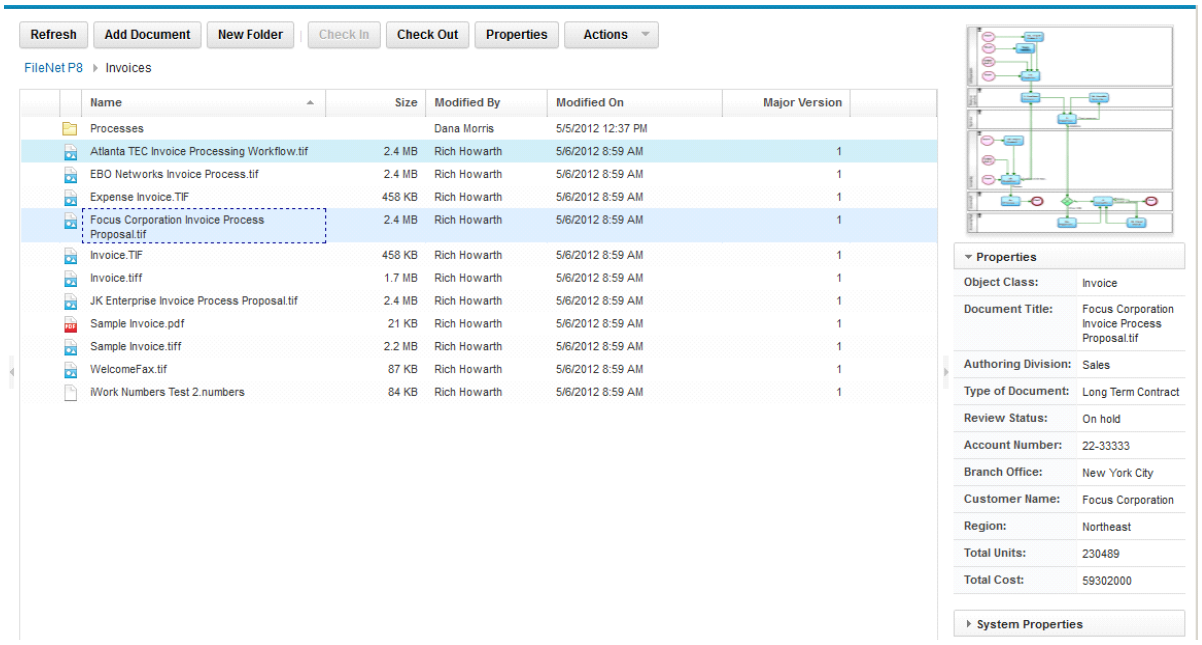
Enter http://Base-Win2k8x64:9080/navigator?desktop=ECM and hit Enter.

Enter **filenetadmin** as the user and **IBMFileNetP8** as the password.

After logging in, select the **Browse** icon on the left hand side of the screen.

Expand the **Invoices** folder.

Select any document in the list. Notice that the label now says **Object Class** in the **Properties** section on the right hand side of the Browser.



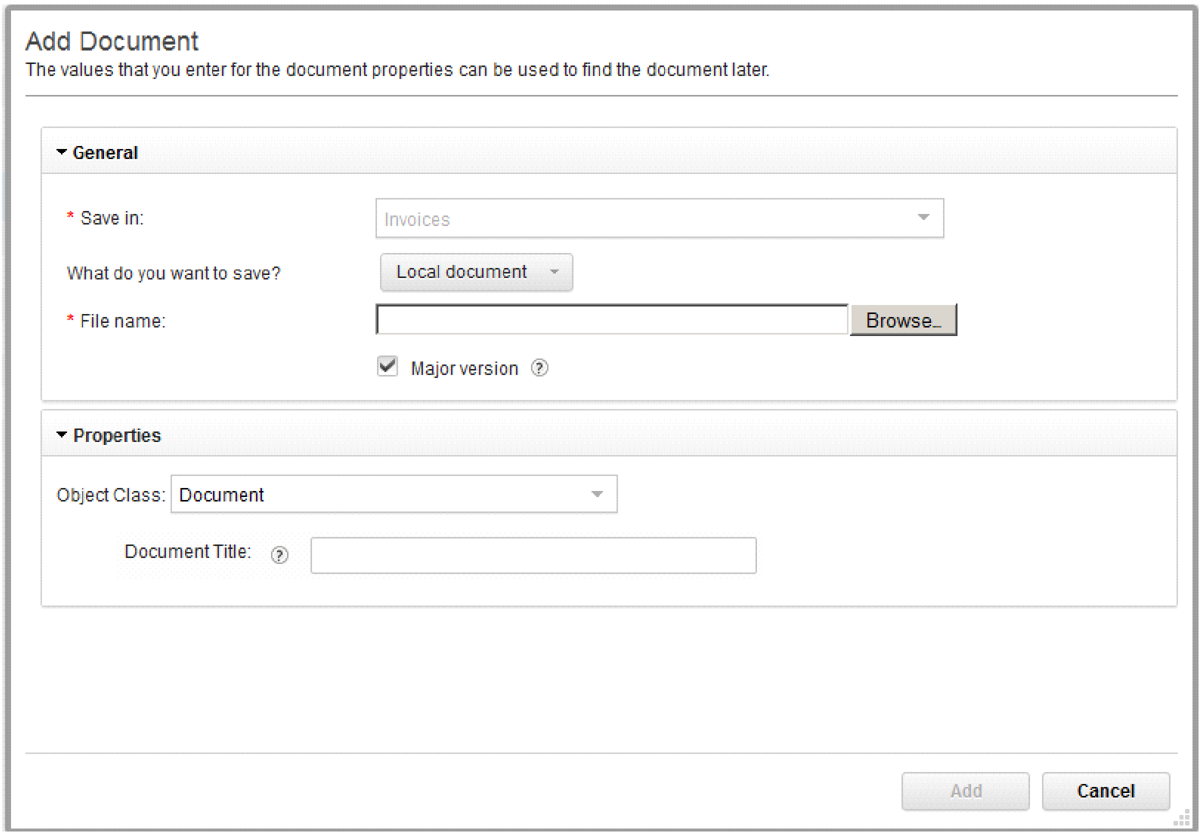
Name	Size	Modified By	Modified On	Major Version
Processes		Dana Morris	5/5/2012 12:37 PM	
Atlanta TEC Invoice Processing Workflow.tif	2.4 MB	Rich Howarth	5/6/2012 8:59 AM	1
EBO Networks Invoice Process.tif	2.4 MB	Rich Howarth	5/6/2012 8:59 AM	1
Expense Invoice.TIF	458 KB	Rich Howarth	5/6/2012 8:59 AM	1
Focus Corporation Invoice Process Proposal.tif	2.4 MB	Rich Howarth	5/6/2012 8:59 AM	1
Invoice.TIF	458 KB	Rich Howarth	5/6/2012 8:59 AM	1
Invoice.tiff	1.7 MB	Rich Howarth	5/6/2012 8:59 AM	1
JK Enterprise Invoice Process Proposal.tif	2.4 MB	Rich Howarth	5/6/2012 8:59 AM	1
Sample Invoice.pdf	21 KB	Rich Howarth	5/6/2012 8:59 AM	1
Sample Invoice.tiff	2.2 MB	Rich Howarth	5/6/2012 8:59 AM	1
WelcomeFax.tif	87 KB	Rich Howarth	5/6/2012 8:59 AM	1
iWork Numbers Test 2.numbers	84 KB	Rich Howarth	5/6/2012 8:59 AM	1

▾ Properties

Object Class:	Invoice
Document Title:	Focus Corporation Invoice Process Proposal.tif
Authoring Division:	Sales
Type of Document:	Long Term Contract
Review Status:	On hold
Account Number:	22-33333
Branch Office:	New York City
Customer Name:	Focus Corporation
Region:	Northeast
Total Units:	230489
Total Cost:	59302000

▸ System Properties

Next click the **Add Document** button. Notice that the label here also now says **Object Class.**



Add Document

The values that you enter for the document properties can be used to find the document later.

▾ General

* Save in:	Invoices
What do you want to save?	Local document
* File name:	
✓ Major version	

▾ Properties

Object Class:	Document
Document Title:	

Lab Activity 5 –External Data Services

In this chapter we introduce a usage scenario leveraging the capabilities that are provided by the External Data Service (EDS) which enables IBM Content Navigator to load data dynamically from external sources such as database tables, JSON files, or any other source of data you have in your enterprise. Some examples of the how you can integrate the data include managing property behavior by prefilling some property values, populating choice lists, setting a field as required, and validating property values input by the user in real-time.

An Example Scenario

The insurance company handles property, casualty, and auto policies. All documents relating to policies are stored in the insurance company's Enterprise Content Management (ECM) repository.
Inbound correspondence is scanned, indexed, and stored in a folder-based filing system by document type within a policy number of a unique client number.
The filing of inbound documentation is dependent upon the assignment of correct values to the inbound document. An invalid or misspelled word means that the document will not be filed correctly. Incorrectly filed documents do not enter the appropriate business process leading to delays and additional costs.
In this scenario we'll use external data services to manage the consistency of data entered during the post scanning indexing phase of in bound document capture.
There are several use cases that can be met by leveraging EDS capabilities:

Provide a choice list for all the estimators for a car insurance claim. This can be a list of people stored in a database table external to the IBM ECM system.

Provide a dependent choice list of categories belonging to one main category. This is to limit the available categories underneath each main category. Some example categories might be regions, with the subcategories being cities with offices.

Validate entered claim numbers by format, e.g. if they contain the exact number and type of characters.

Hide input property field to enter a custom text for the reason an insurance claim was opened if none of the provided ones out of the choice list would make sense.

Sample Project Description

In the next few sections we will be creating a updating the sample EDS implementation that is provided with IBM Content Navigator. The sample EDS implementation can be found under the path: **C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\sampleEDSService_war.ear\**

The sample EDS service provided with Content Navigator is a simple web application that contains two Java servlet classes. The two servlet classes are **UpdateObjectTypeServlet** and **GetObjectTypesServlet**. The **GetObjectTypesServlet** provides an HTTP GET service to determine the list of object types supported by this EDS implementation. The **UpdateObjectTypeServlet** provides the HTTP POST service that is used to obtain external data and dynamically change, validate and enrich metadata in real-time.

The rest of the files that are contained in the sample are JavaScript Object Notation (JSON) files holding the data returned by this EDS implementation. **ObjectTypes.json** is the key file that describes which FileNet P8 Object classes in the IBM ECM repository should the EDS service be expected to work with. The other JSON files are sample JSON files that match to some of the Object class names in P8. These files contain the sample data that will manipulate the property values shown in Content Navigator.

For the rest of this chapter, we'll focus on a simple scenario where you will add data validation and external choice lists to the Presentation object class. The Presentation class will be validating and adding data based on session information. Please keep in mind that the sample EDS service is set up to provide EDS using "hard-coded" JSON files. In a more real-world scenario, such as the one covered in the Redbook SG248055 "Customizing and Extending Content Navigator", the services would more typically be a remote web service that returns data programmatically based on calls to a relational database or 3rdParty system.

For all of the subsequent walkthroughs, we'll be making changes to the sample EDS service which is already deployed on the image. You can find the JSON files for this service in the following directory:

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

Data Validation

In this section we'll walk through an example of adding property validation.

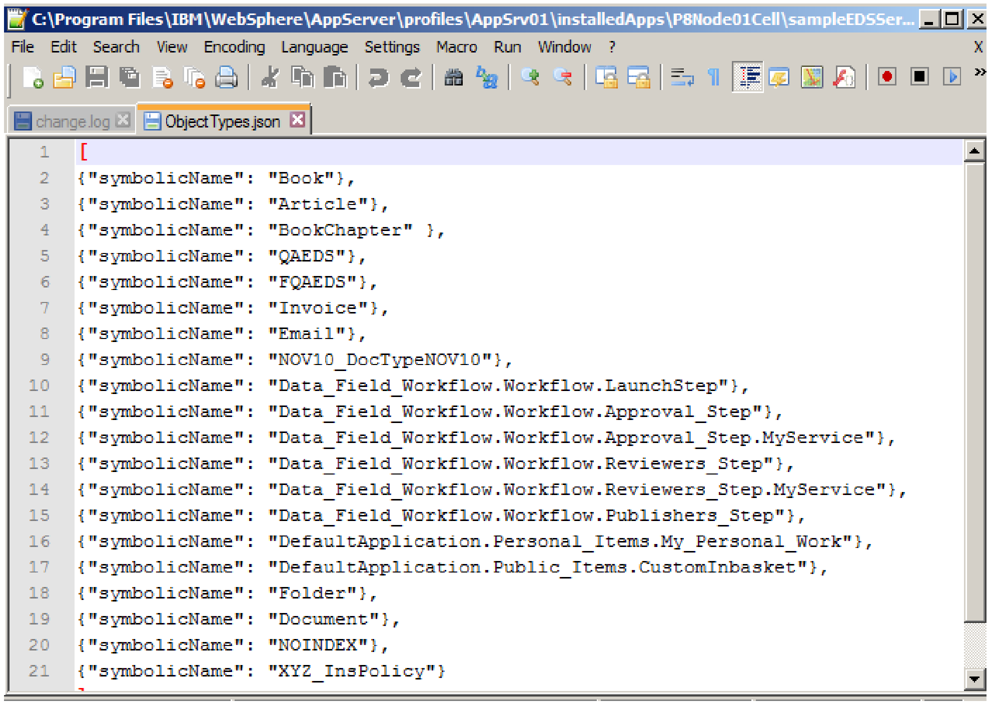
Step 1 – Update ObjectTypes.json

First, we need to update the sample ObjectTypes.json file to know that it should expect to return information anytime the Presentation object class is referenced in the Content Navigator application.

Open Windows Explorer

Navigate to the deployed EDS service directory at **C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\sampleEDSService_war.ear\sampleEDSService.war\WEB-INF\classes**

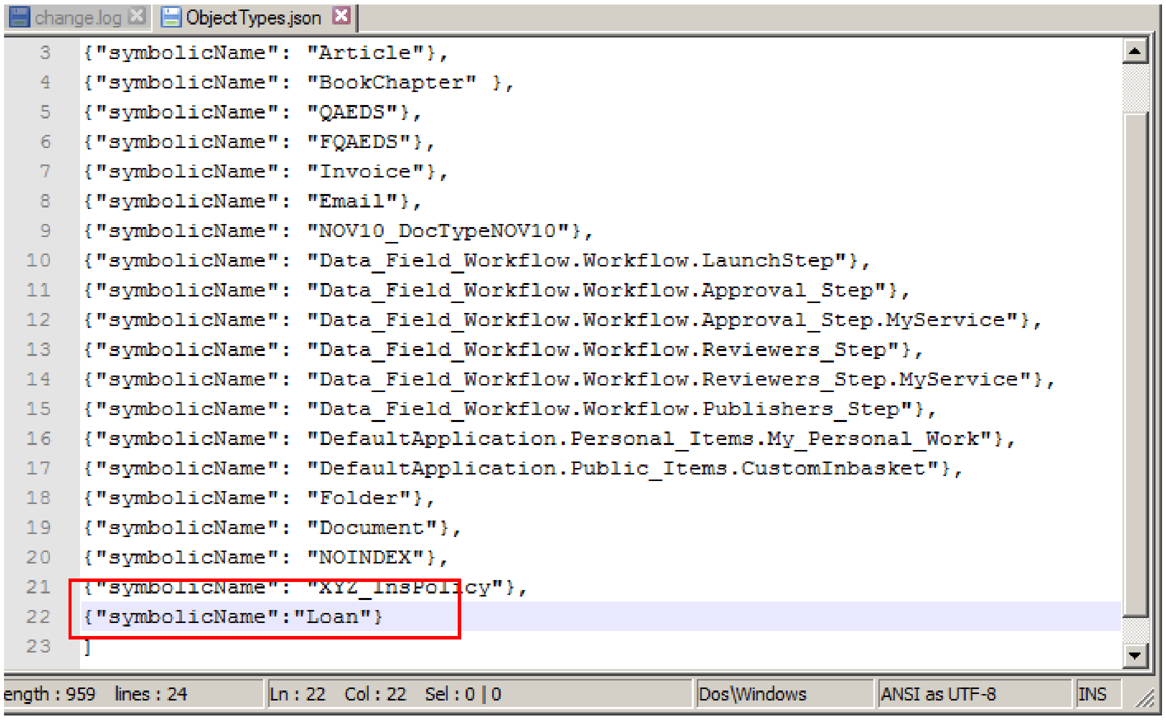
Open the **ObjectTypes.json** file in Notepad. It should look something like the following.


C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\sampleEDSSer...
File Edit Search View Encoding Language Settings Macro Run Window ?
change.log
ObjectTypes.json

```
[
{"symbolicName": "Book"},
{"symbolicName": "Article"},
{"symbolicName": "BookChapter" },
{"symbolicName": "QAEDS"},
{"symbolicName": "FQAEDS"},
{"symbolicName": "Invoice"},
{"symbolicName": "Email"},
{"symbolicName": "NOV10_DocTypeNOV10"},
{"symbolicName": "Data_Field_Workflow.Workflow.LaunchStep"},
{"symbolicName": "Data_Field_Workflow.Workflow.Approval_Step"},
{"symbolicName": "Data_Field_Workflow.Workflow.Approval_Step.MyService"},
{"symbolicName": "Data_Field_Workflow.Workflow.Reviewers_Step"},
{"symbolicName": "Data_Field_Workflow.Workflow.Reviewers_Step.MyService"},
{"symbolicName": "Data_Field_Workflow.Workflow.Publishers_Step"},
{"symbolicName": "DefaultApplication.Personal_Items.My_Personal_Work"},
{"symbolicName": "DefaultApplication.Public_Items.CustomInbasket"},
{"symbolicName": "Folder"},
{"symbolicName": "Document"},
{"symbolicName": "NOINDEX"},
{"symbolicName": "XYZ_InsPolicy"}
```

Now find the last line with a value of **{"symbolicName": "XYZ_InsPolicy"}** and append a comma on the end and then hit Enter.

Add the text **{"symbolicName": "Loan"}** to the file. The file should now look like the following screen.



```
{"symbolicName": "Article"},
{"symbolicName": "BookChapter" },
{"symbolicName": "QAEDS"},
{"symbolicName": "FQAEDS"},
{"symbolicName": "Invoice"},
{"symbolicName": "Email"},
{"symbolicName": "NOV10_DocTypeNOV10"},
{"symbolicName": "Data_Field_Workflow.Workflow.LaunchStep"},
{"symbolicName": "Data_Field_Workflow.Workflow.Approval_Step"},
{"symbolicName": "Data_Field_Workflow.Workflow.Approval_Step.MyService"},
{"symbolicName": "Data_Field_Workflow.Workflow.Reviewers_Step"},
{"symbolicName": "Data_Field_Workflow.Workflow.Reviewers_Step.MyService"},
{"symbolicName": "Data_Field_Workflow.Workflow.Publishers_Step"},
{"symbolicName": "DefaultApplication.Personal_Items.My_Personal_Work"},
{"symbolicName": "DefaultApplication.Public_Items.CustomInbasket"},
{"symbolicName": "Folder"},
{"symbolicName": "Document"},
{"symbolicName": "NOINDEX"},
{"symbolicName": "XYZ_InsPolicy"},
{"symbolicName":"Loan"}
]
```

Save and close the file.

You've now updated the EDS service to instruct the sample Java servlet to respond to any request from Content Navigator for data for the Object class with the symbolic name of "**Loan**".

Step 2 – Add Validation for the Session Number

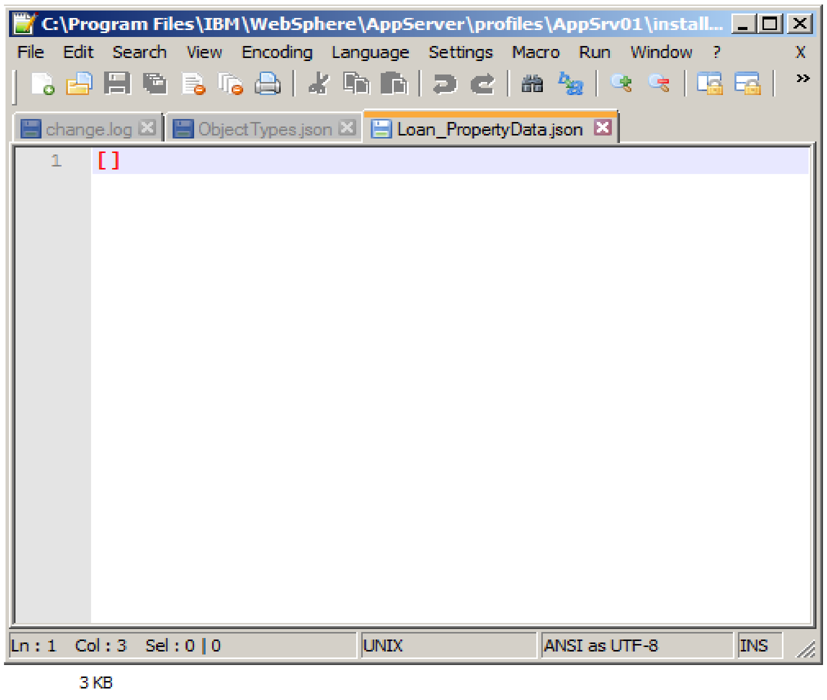
Next, we'll add data validation for the **LoanNumber** property of the **Loan** class.

Open Windows Explorer

Navigate to the deployed EDS service directory at **C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01 Cell\sampleEDSService_war.ear\sampleEDSService.war\WEB-INF\classes**

Open the Loan_**PropertyData.json** file in Notepad. It should look something like the following.

NOTE: If you are going to copy and paste the below lines from the pdf, make sure you retype the open and close quotation marks. PDF will put junk characters and will result in errors in the EDS


C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\install...
File Edit Search View Encoding Language Settings Macro Run Window ?
change.log
ObjectTypes.json
Loan_PropertyData.json

```
[]
```

Ln : 1 Col : 3 Sel : 0 | 0
UNIX
ANSI as UTF-8
INS
3 KB

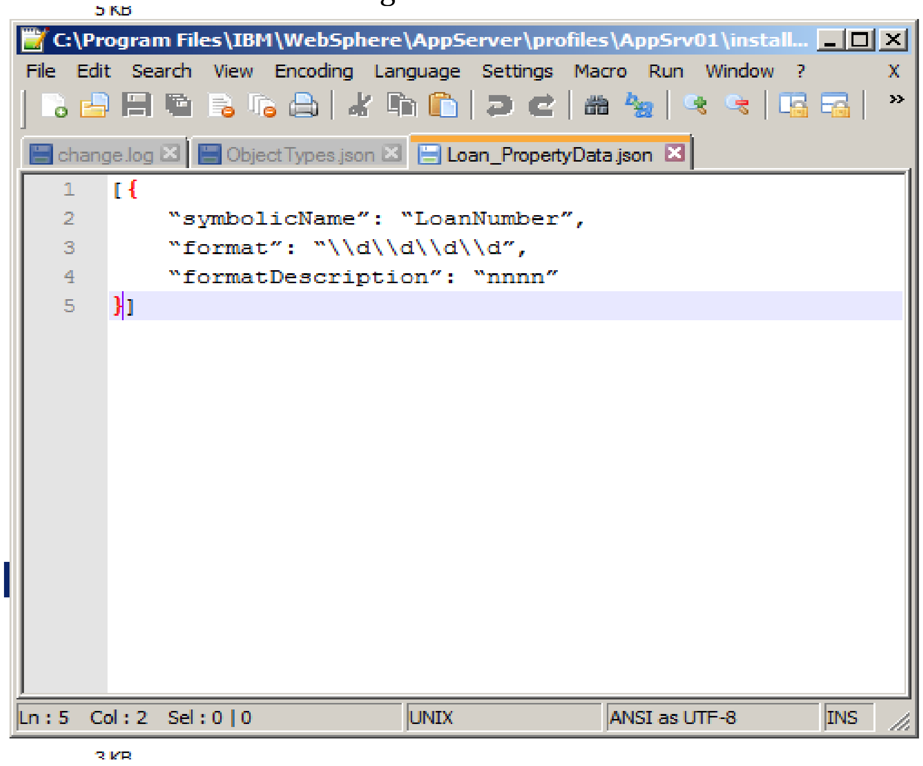
After the **[** character hit Enter to insert a new line.

Add the following text to the file.

{

```
            "symbolicName": "LoanNumber",
            "format": "\\d\\d\\d\\d",
            "formatDescription": "nnnn"
    }
```

The file should now look like the following screen.



C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\install...
File Edit Search View Encoding Language Settings Macro Run Window ?
change.log
ObjectTypes.json
Loan_PropertyData.json
Ln : 5 Col : 2 Sel : 0 | 0
UNIX
ANSI as UTF-8
INS

```
[{
    "symbolicName": "LoanNumber",
    "format": "\\d\\d\\d\\d",
    "formatDescription": "nnnn"
}]
```

Save and close the file.

What you've just done is to add some very simple validation for the **LoanNumber** property of the **Loan** class. In our example, Loan numbers must be 4 digits long so we've added a regular expression to validate that the input value is exactly 4 digits long with no spaces, special characters, or text characters allowed. The **formatDescription** is the text that will be displayed to the user when they input an invalid value. That is basically the value that will be displayed to help the user know what format is expected.

Step 3 – Restart the sampleEDSService Web Application

Next, you'll log in to the WebSphere Administration console to restart the deployed sampleEDSService application. This will then force the application to reload the JSON files so that they will be applied when we next log in to Content Navigator.

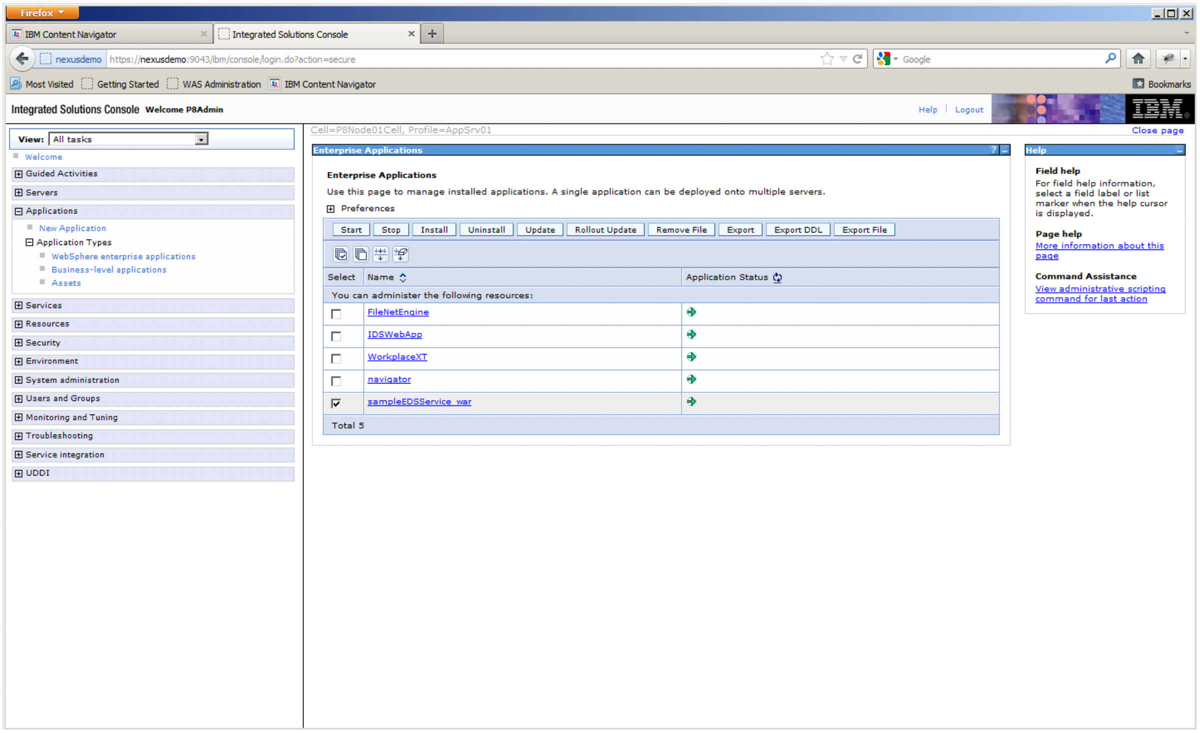
Double click on the **Administrative Console** icon on the desktop to launch the WebSphere Application Server administration.

Enter **p8admin** as the user and **IBMFileNetP8** for the password.

Expand **Applications** and then **Application Types**. Click on the **WebSphere enterprise applications** link.

Select the **sampleEDSService** application and then click the **Stop** button.

Select the **sampleEDSService** application and then click the **Start** button.



https://nexusdemo:9043/ibm/console/login.do?action=secure

Integrated Solutions Console Welcome P8Admin

Cell=P8Node01Cell, Profile=AppSrv01

Enterprise Applications

Use this page to manage installed applications. A single application can be deployed onto multiple servers.

Select	Name	Application Status
You can administer the following resources:		
☐	FileNetEngine	➜
☐	IDSWebApp	➜
☐	WorkplaceXT	➜
☐	navigator	➜
✓	sampleEDSService_war	➜
Total 5		

Help

Field help
For field help information, select a field label or list marker when the help cursor is displayed.

Page help
More information about this page

Command Assistance
View administrative scripting command for last action

Click the **Logout** link in the upper right corner of the browser window.

Close IE

Step 4 – Test the EDS Service

Next we'll log in to Content Navigator and test out the changes we've just made by trying to add a document using the Presentation class.

Open IE by clicking on the icon in the Windows quick launch bar.

Enter http://Base-Win2k8x64:9080/navigator?desktop=ECM as the URL and hit **Enter**.

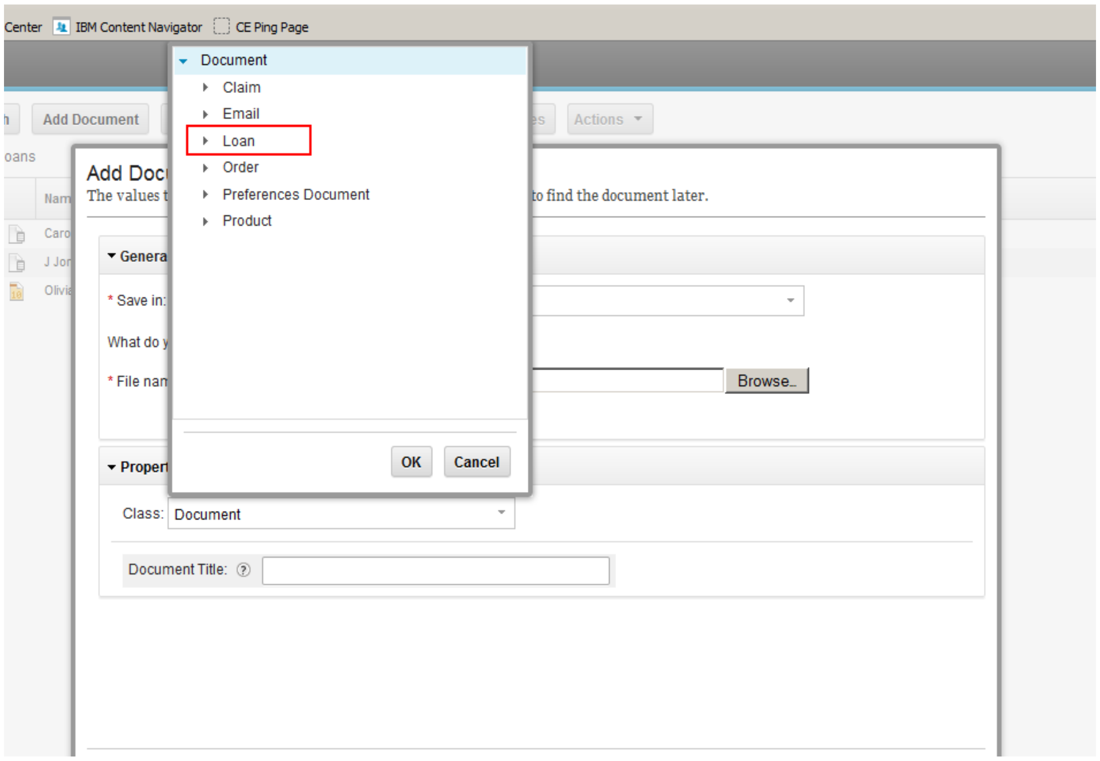
Log in using **filenetadmin** for the user and **IBMFileNetP8** for the password.

Click on the **Browse** icon on the left hand side of the screen.

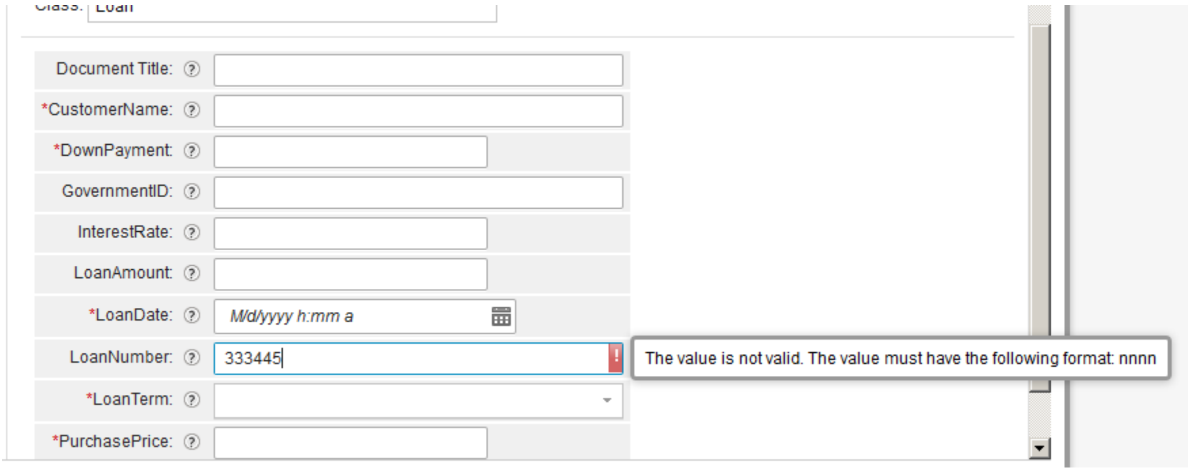
Open the **Loans** folder.

Click the **Add Document** button.

In the Object Class drop down select **Loan** and click OK.



Next click on the Loan **Number** text box. Enter the letter **f** or any other text character. You should see the field change to being marked with a red * and the help text should tell you that the value you have entered is invalid.



LoanNumber: 333445

The value is not valid. The value must have the following format: nnnn

Try correcting the data but putting in 5 digits. You should see that an error occurs because there are too many digits.

Close IE.

Choice Lists

In this section we'll walk through an example of adding external choice lists. Choice lists are a great way to help prevent indexing errors because they limit the choices available to the user and they are pre-configured so that users cannot make any mistakes typing in data.

Step 1 – Add a Choice List for the Loan Term

Next, we'll add a choice list for the LoanTerm property of the Loan class.

Open Windows Explorer

Navigate to the deployed EDS service directory at **C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\sampleEDSService_war.ear\sampleEDSService.war\WEB-INF\classes**

Open the Loan**_PropertyData.json** file in Notepad.

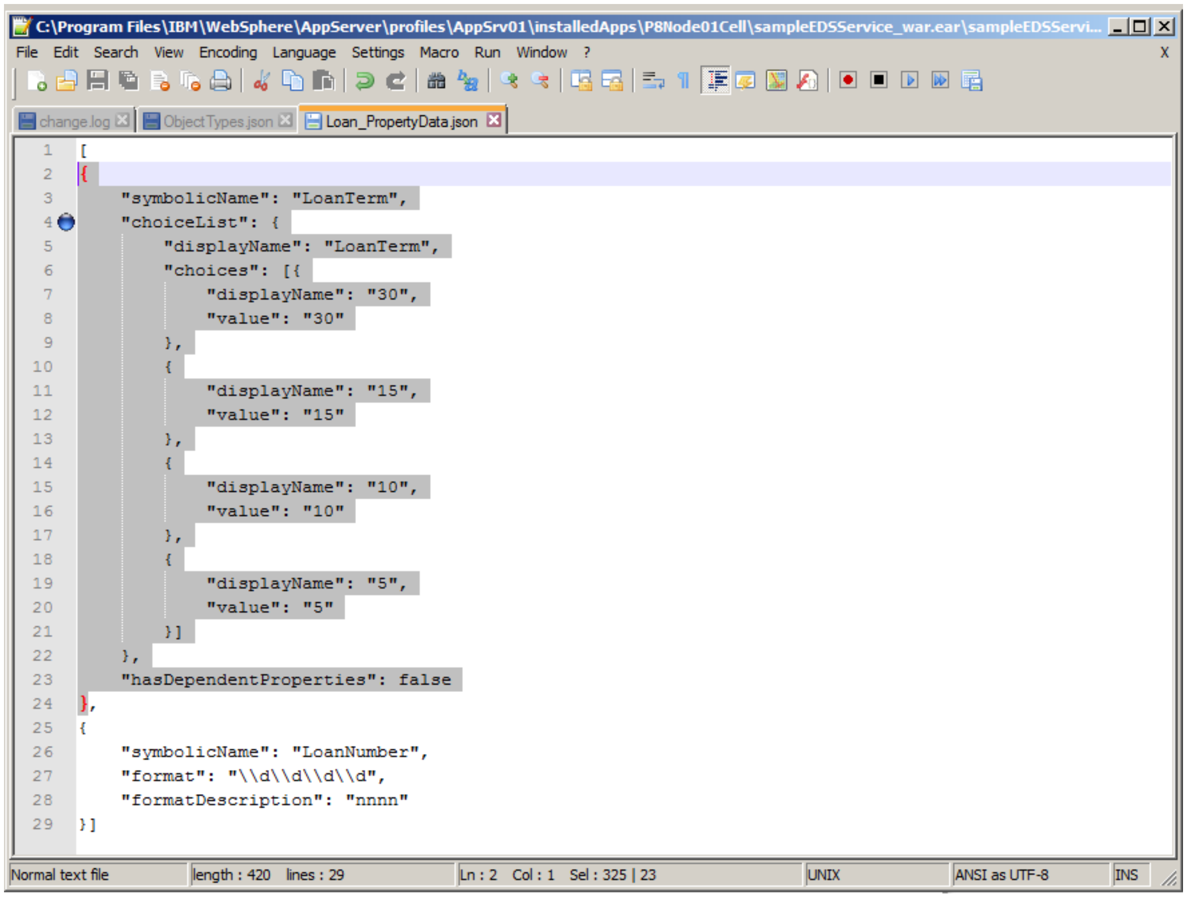
After the **[** character hit Enter to insert a new line.

NOTE: If you are going to copy and paste the below lines from the pdf, make sure you retype the open and close quotation marks. PDF will put junk characters and will result in errors in the EDS

Add the following text to the file.

```
{
        "symbolicName": "LoanTerm",
        "choiceList": {
                "displayName": "LoanTerm",
                "choices": [{
                        "displayName": "30",
                        "value": "30"
                },
                {
                        "displayName": "15",
                        "value": "15"
                },
                {
                        "displayName": "10",
                        "value": "10"
                },
                {
                        "displayName": "5",
                        "value": "5"
                }]
        },
        "hasDependentProperties": false
},
```

The file should now look like the following screen.


C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\sampleEDSService_war.ear\sampleEDSServi...
File Edit Search View Encoding Language Settings Macro Run Window ?
change.log
ObjectTypes.json
Loan_PropertyData.json
Normal text file
length : 420 lines : 29
Ln : 2 Col : 1 Sel : 325 | 23
UNIX
ANSI as UTF-8
INS

```
[
{
    "symbolicName": "LoanTerm",
    "choiceList": {
        "displayName": "LoanTerm",
        "choices": [{
            "displayName": "30",
            "value": "30"
        },
        {
            "displayName": "15",
            "value": "15"
        },
        {
            "displayName": "10",
            "value": "10"
        },
        {
            "displayName": "5",
            "value": "5"
        }]
    },
    "hasDependentProperties": false
},
{
    "symbolicName": "LoanNumber",
    "format": "\\d\\d\\d\\d",
    "formatDescription": "nnnn"
}]
```

Save and close the file.

What you've just done is to add a very simple list of 4 choices from which the user can select a single value from to assign to the **LoanTerm** property.

Step 2 – Restart the sampleEDSService Web Application

Next, you'll log in to the WebSphere Administration console to restart the deployed sampleEDSService application. This will then force the application to reload the JSON files so that they will be applied when we next log in to Content Navigator.

Double click on the **Administrative Console** icon on the desktop to launch the

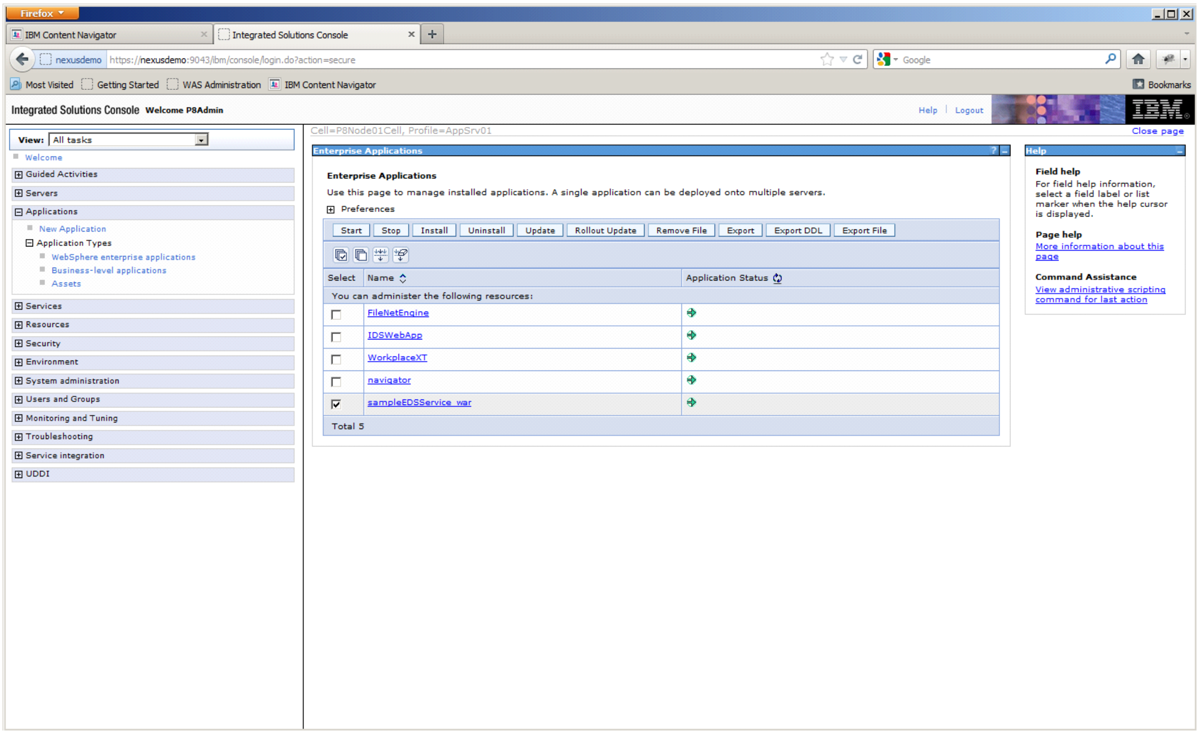
WebSphere Application Server administration.

Enter **p8admin** as the user and **IBMFileNetP8** for the password.

Expand **Applications** and then **Application Types**. Click on the **WebSphere enterprise applications** link.

Select the **sampleEDSService** application and then click the **Stop** button.

Select the **sampleEDSService** application and then click the **Start** button.



https://nexusdemo:9043/ibm/console/login.do?action=secure

Integrated Solutions Console Welcome P8Admin

Cell=P8Node01Cell, Profile=AppSrv01

Enterprise Applications

Enterprise Applications

Use this page to manage installed applications. A single application can be deployed onto multiple servers.

⊞ Preferences

Select	Name	Application Status
You can administer the following resources:		
☐	FileNetEngine	➔
☐	IDSWebApp	➔
☐	WorkplaceXT	➔
☐	navigator	➔
☑	sampleEDSService_war	➔
Total 5		

Help

Field help
For field help information, select a field label or list marker when the help cursor is displayed.

Page help
More information about this page

Command Assistance
View administrative scripting command for last action

Click the **Logout** link in the upper right corner of the browser window.

Close IE

Step 3 – Test the EDS Service

Next we'll log in to Content Navigator and test out the changes we've just made by trying to add a document using the Presentation class.

Open IE by clicking on the icon in the Windows quick launch bar.

Enter **http://Base-Win2k8x64:9080/navigator?desktop=**ECM as the URL and hit **Enter**.

Log in using **filenetadmin** for the user and **IBMFileNetP8** for the password.

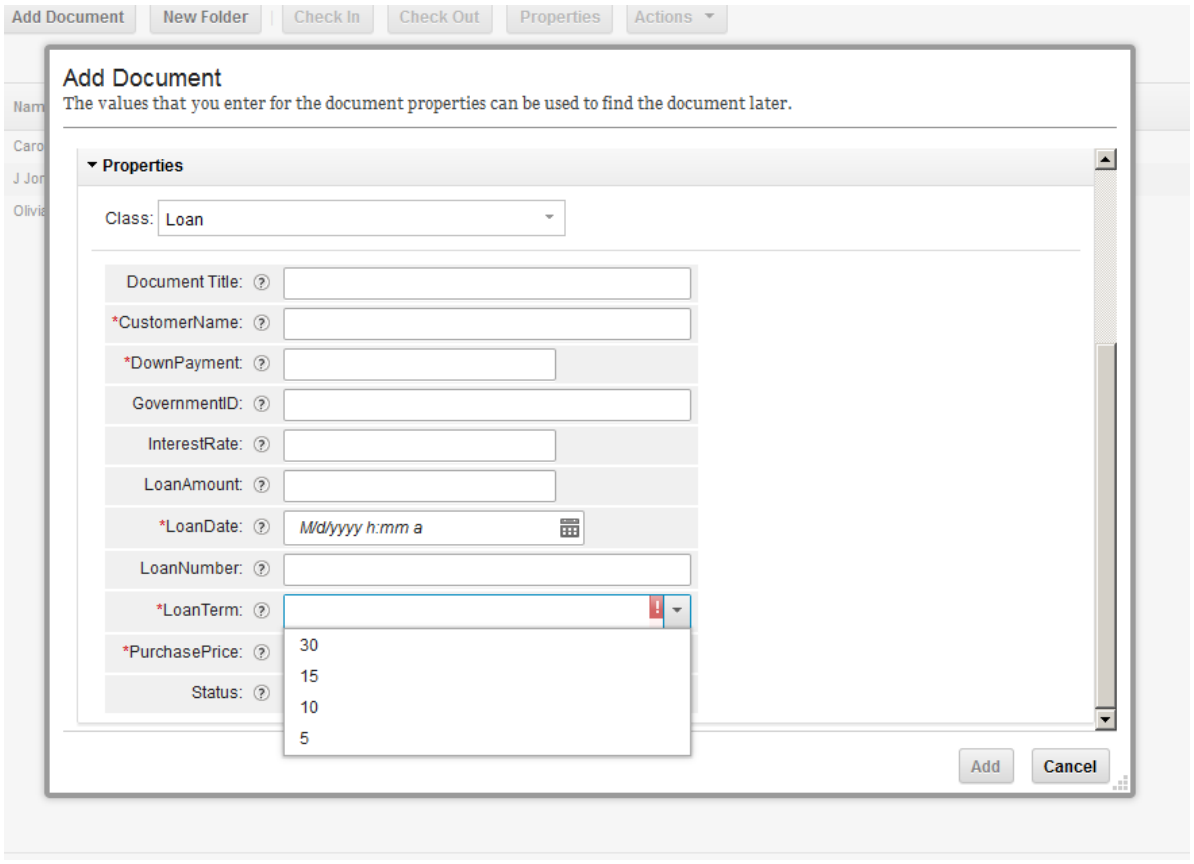
Click on the **Browse** icon on the left hand side of the screen.

Open the **Loans** folder.

Click the **Add Document** button.

In the Object Class drop down select **Loan** and click OK.

Next click on the Loan **Term** property. This property should now have a down arrow allowing you to select from the 4 choices you entered in step 1.



Close IE.

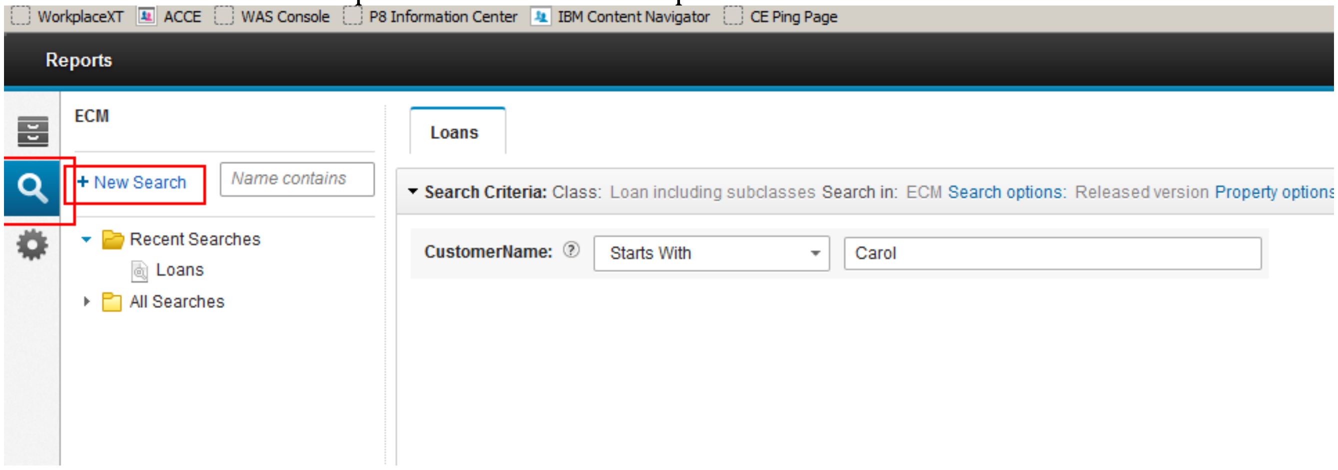
Dependent Choice Lists

In this section we're not going to specifically add any additional EDS customizations. For an example of how to set up a Choice List dependency, you can open up the Invoice_PropertyData.json file in the . **C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\P8Node01Cell\sampleEDSService.ear\sampleEDSService.war\WEB-INF\classes** directory. The specific example of dependent choice lists is between the **Region** and **BranchOffice** properties. When the user selects a **Region**, the **BranchOffice** choice list is updated to reflect choices that are valid for the given region. Try adding a document to the **Invoice** object class to see an example of this powerful capability in action.
There is also additional information in the Content Navigator programming guide that details the other capabilities of Content Navigator's EDS implementation.

Lab Activity 5 – Search

To create a new search, complete the following steps:

1. Click the Search view icon (it looks like a magnifying glass) if you do not already have the Search Criteria pane open.
2. Click New Search to open the Search Criteria pane.



Search Criteria: Class: Loan including subclasses Search in: ECM Search options: Released version Property options

CustomerName: Starts With Carol

3. Click Class at the top left of the Search Criteria pane.
If you select a specific document class, you can sort through all of the content in the repository but limit the search to a certain document class. For example, if you want to search and find only loans, you select Loan from the document class list; the search results show only documents that are a part of the Loan document class.

Lab Activity 6 – Developing Plug-ins

A Content Navigator plug-in allows you extend the product to meet any specific business needs. By creating a plug-ins, you can add functionality directly into the Content Navigator user interface, you can extend existing functionality, or you can completely rework the application.
A Content Navigator plug-in must implement a set of abstract Java classes that provide Content Navigator with information on the functionality provided by the plug-in as well as how that functionality should be integrated with the base product.
The code for a plug-in is packaged as a single JAR file that is then registered with Content Navigator through the Content Navigator administration. After the plug-in is registered with Content Navigator, the functionality provided by the plug-in will be available from within the base product.
A plug-in can extend and enhance virtually any aspect of the Content Navigator experience. You can create a plug-in to provide any of the following extensions:

A custom action (ex: Preview or Download as PDF)

An additional menu (ex: IBM Enterprise Records records declaration)

Entirely new features (ex: IBM Content Analytics)

Custom services (ex: IBM Content Analytics)

Additional viewers (ex: several IBM Business Partners provide custom viewers that can be integrated)

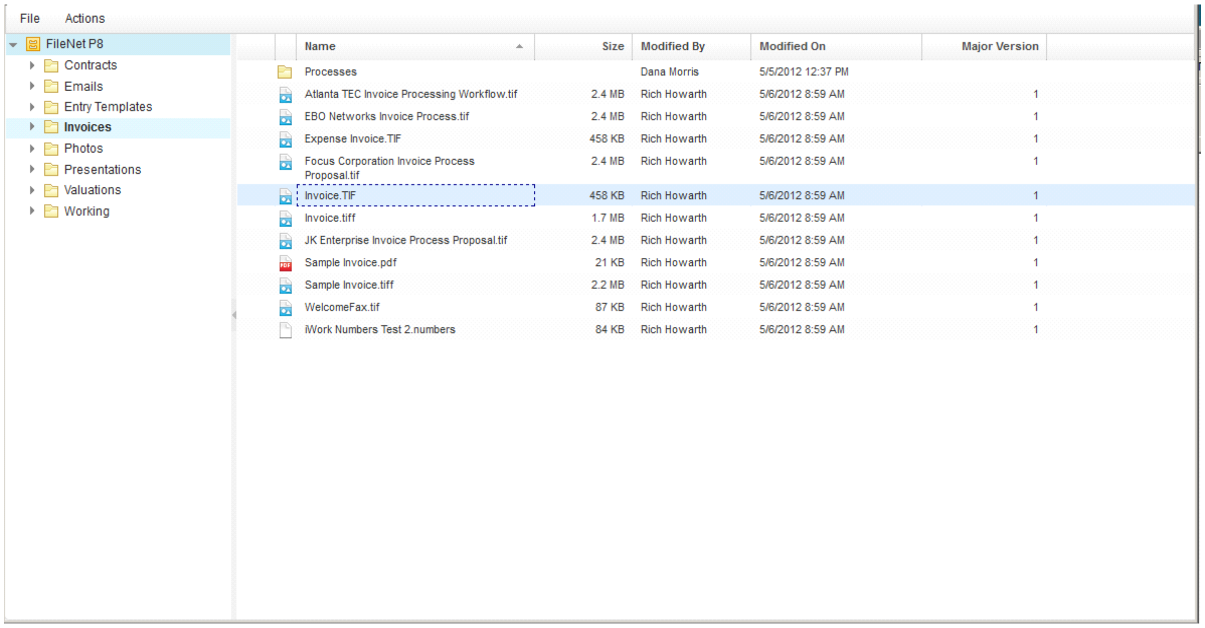
An entirely new layout

Intercept and modify the requests or responses sent (ex: External Data Services)

Custom widgets (ex: the Content Navigator tree control or add document dialog are visual widgets)

Configuration screens for setting up the plugin specific information in the Content Navigator administration

A plugin JAR file is self-contained and can be as simple as a single new action to containing everything needed for an entirely new set of services, widgets, and layout. The contents of a plug-in are always defined through the Plugin class that belongs to every plug-in project.
For more details on any of the above methods of extension, refer to the IBM Content Navigator Programming Guide or the Customizing and Extending IBM Content Navigator redbook.
For the remainder of this exercise, you'll walk through an example of creating and developing a new Plug-in to deploying it in Content Navigator for testing. This particular plugin will create a new Microsoft Windows Explorer style layout that will look like the following



File Actions

- FileNet P8
  - Contracts
  - Emails
  - Entry Templates
  - Invoices
  - Photos
  - Presentations
  - Valuations
  - Working

Name	Size	Modified By	Modified On	Major Version
Processes		Dana Morris	5/5/2012 12:37 PM	
Atlanta TEC Invoice Processing Workflow.tif	2.4 MB	Rich Howarth	5/6/2012 8:59 AM	1
EBO Networks Invoice Process.tif	2.4 MB	Rich Howarth	5/6/2012 8:59 AM	1
Expense Invoice.TIF	458 KB	Rich Howarth	5/6/2012 8:59 AM	1
Focus Corporation Invoice Process Proposal.tif	2.4 MB	Rich Howarth	5/6/2012 8:59 AM	1
Invoice.TIF	458 KB	Rich Howarth	5/6/2012 8:59 AM	1
Invoice.tiff	1.7 MB	Rich Howarth	5/6/2012 8:59 AM	1
JK Enterprise Invoice Process Proposal.tif	2.4 MB	Rich Howarth	5/6/2012 8:59 AM	1
Sample Invoice.pdf	21 KB	Rich Howarth	5/6/2012 8:59 AM	1
Sample Invoice.tiff	2.2 MB	Rich Howarth	5/6/2012 8:59 AM	1
WelcomeFax.tif	87 KB	Rich Howarth	5/6/2012 8:59 AM	1
iWork Numbers Test 2.numbers	84 KB	Rich Howarth	5/6/2012 8:59 AM	1

42.1 Creating a new Eclipse Project

The first step is to create a new project in Eclipse. Eclipse is an open-source Java and web development Integrated Development Environment (IDE). We'll be using Eclipse for much of the remainder of this lab.

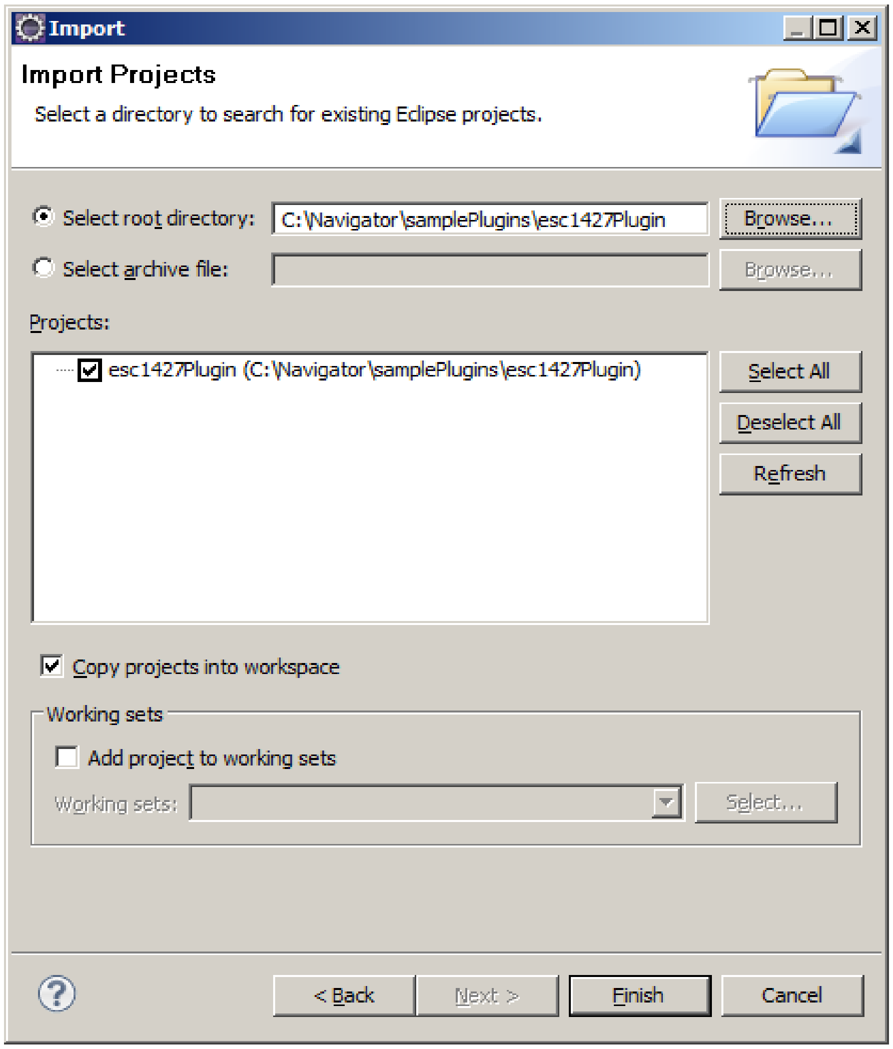
Step 1 – Open the Eclipse IDE

Double-click on the **Eclipse** icon on the desktop.

Step 2 – Create a new Java Project

Next, we'll import an existing project that will set up the basics for us.

1. Click on **File->Import**
2. Expand the General folder and click on **Existing Projects into Workspace**
3. On the next screen click the "**Browse**" button and select the **esc1427Plugin**. (The plugin is in C:\Navigator\sampleplugsin folder)


Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\Navigator\samplePlugins\esc1427Plugin

( ) Select archive file:

Projects:

[✓] esc1427Plugin (C:\Navigator\samplePlugins\esc1427Plugin)

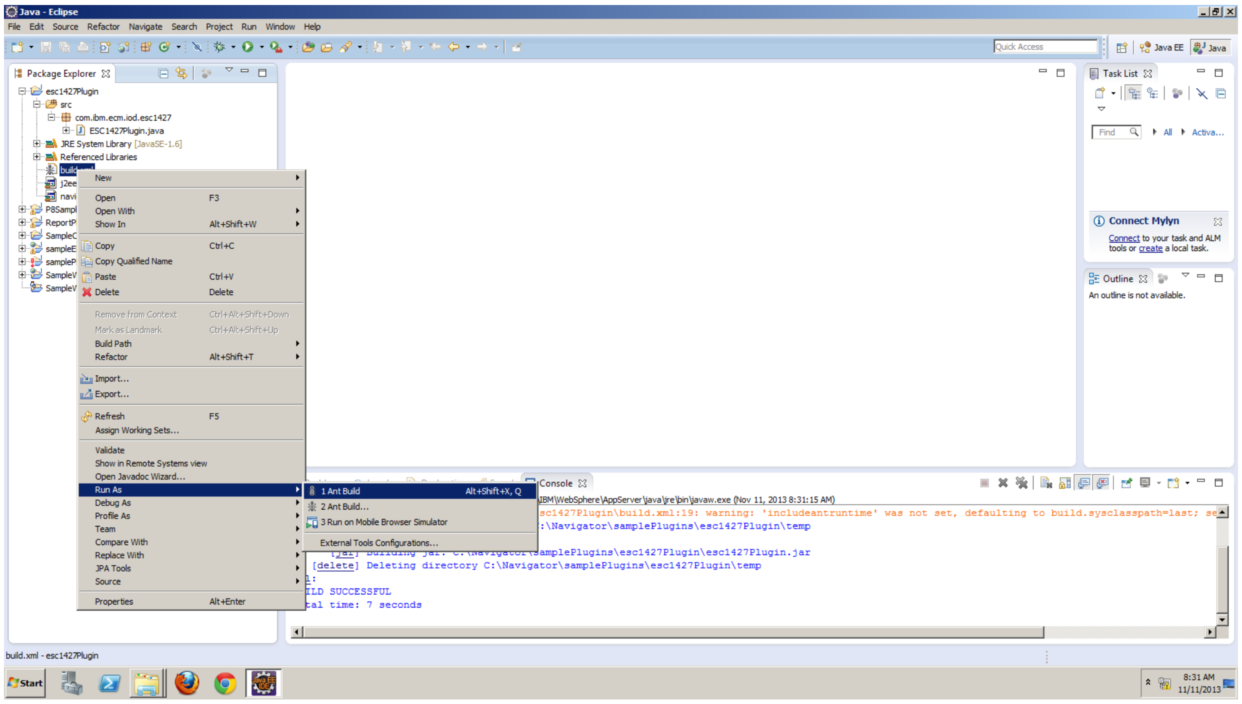
[✓] Copy projects into workspace

Working sets

[ ] Add project to working sets

Working sets:

4. Click **Finish**.
5. Expand the **src** folder and the subfolders.
6. Build the plugin jar file by right clicking on build.xml → Run As → Ant build

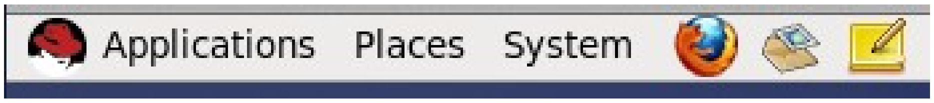


```
IBM\WebSphere\AppServer\java\jre\bin\javaw.exe (Nov 11, 2013 8:31:15 AM)
sc1427Plugin\build.xml:19: warning: 'includeantruntime' was not set, defaulting to build.sysclasspath=last; se
:\Navigator\samplePlugins\esc1427Plugin\temp

amplePlugins\esc1427Plugin\esc1427Plugin.jar
 [delete] Deleting directory C:\Navigator\samplePlugins\esc1427Plugin\temp
l:
ILD SUCCESSFUL
tal time: 7 seconds
```

Step 3 – Register the plug-in in IBM Content Navigator

In IBM Content Navigator 2.0.2 the plug-in administration now 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 had to package your plug-in as a JAR file prior to registering it in IBM Content Navigator. For this exercise, we will register this new plug-in now.

1. Click on the Firefox icon in the top bar:

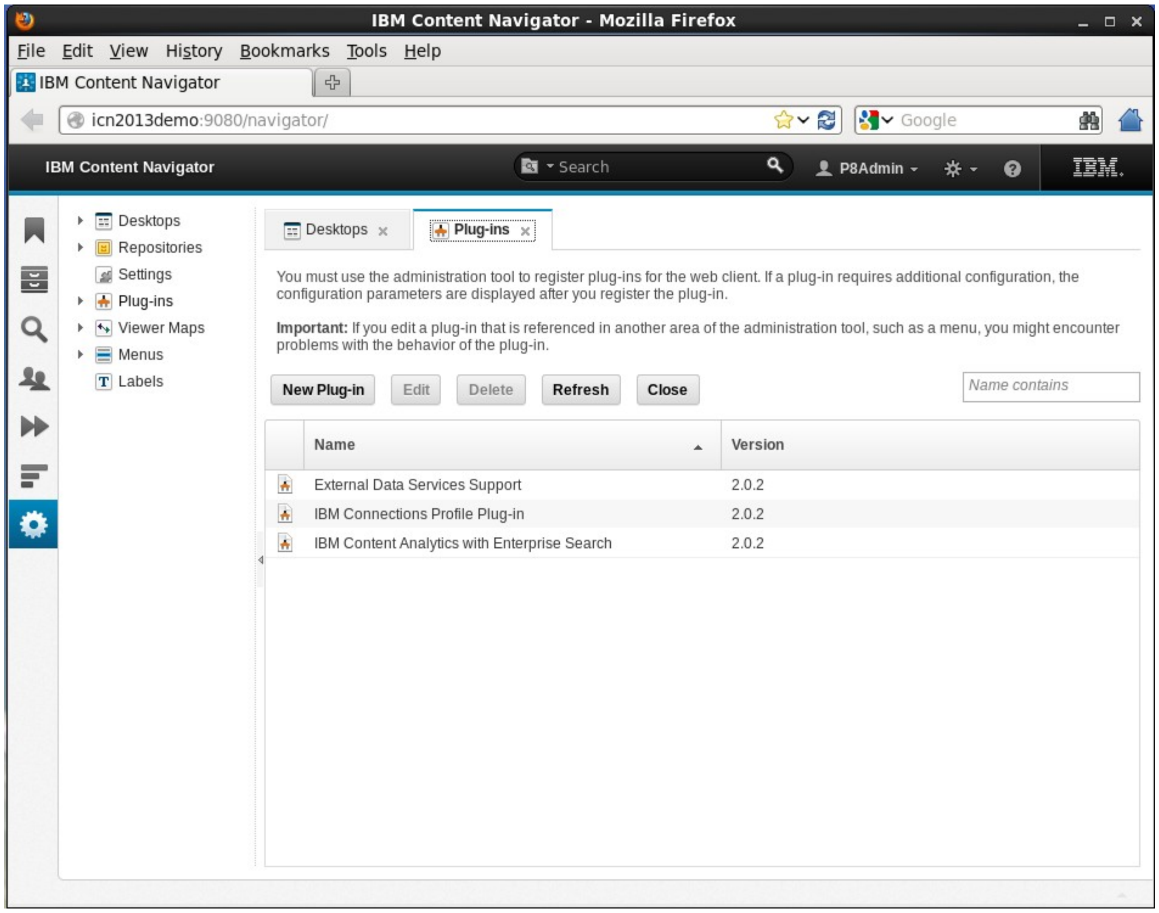


2. In FireFox, select "Bookmarks" and click on "IBM Content Navigator"

3. Login to IBM Content Navigator with user "**filenetadmin**" and password "**IBMFileNetP8**".

4. Click on the "**Administration**" feature, which is the "gear" icon at the bottom of the navigation bar on the left side of the IBM Content Navigator User Interface.

5. Click on "**Plugins**":



You must use the administration tool to register plug-ins for the web client. If a plug-in requires additional configuration, the configuration parameters are displayed after you register the plug-in.

Important: If you edit a plug-in that is referenced in another area of the administration tool, such as a menu, you might encounter problems with the behavior of the plug-in.

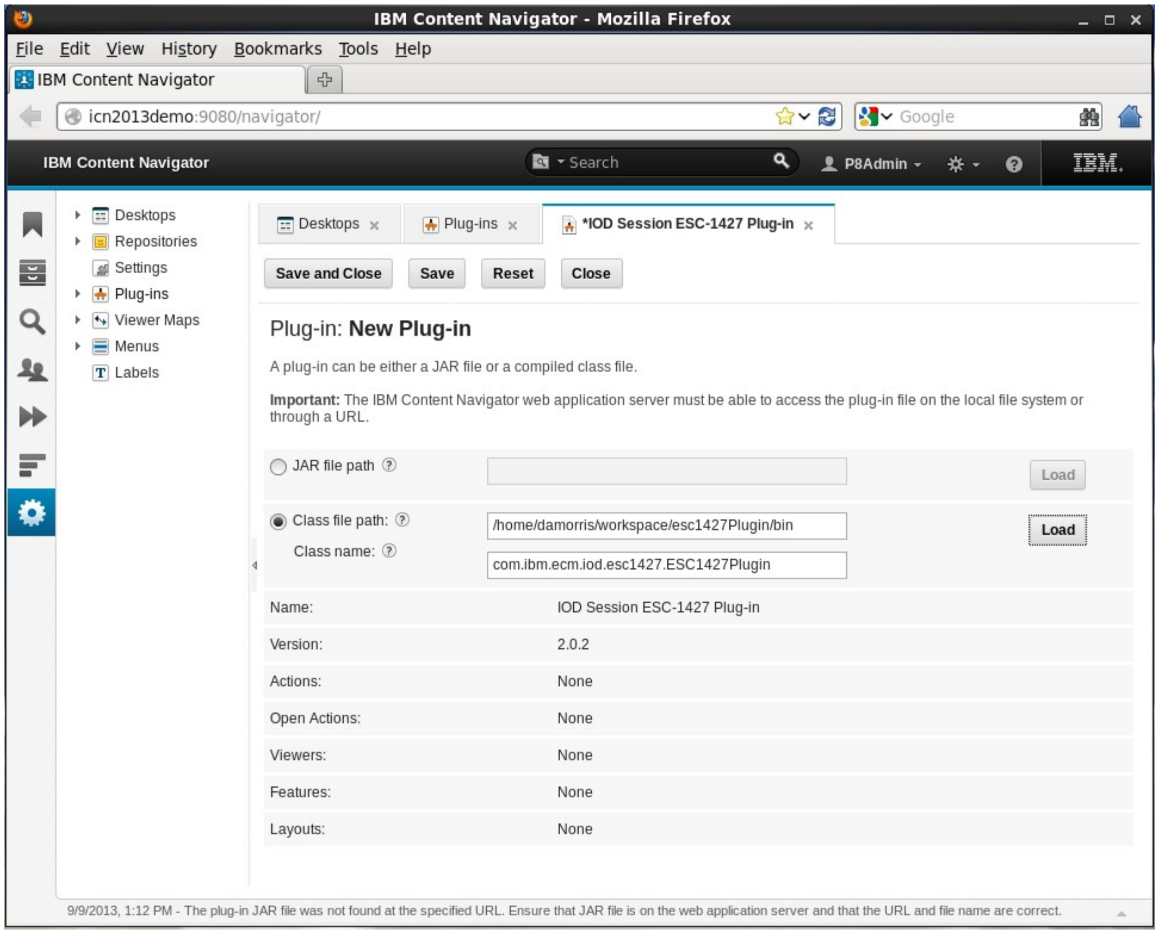
Name	Version
External Data Services Support	2.0.2
IBM Connections Profile Plug-in	2.0.2
IBM Content Analytics with Enterprise Search	2.0.2

6. Click on "**New Plug-in**"

7. On the "**New Plug-in**" screen, select the radio option called "**Class file path**".

8. Enter "**C:\Users\Administrator\workspace\esc1427Plugin\bin**" for the "Class file path" and "**com.ibm.ecm.iod.esc1427.ESC1427Plugin**" for the "Class name".

9. Click the "**Load**" button:


IBM Content Navigator - Mozilla Firefox

icn2013demo:9080/navigator/

Plug-in: **New Plug-in**

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:

Class file path: /home/damorris/workspace/esc1427Plugin/bin

Class name: com.ibm.ecm.iod.esc1427.ESC1427Plugin

Name:	IOD Session ESC-1427 Plug-in
Version:	2.0.2
Actions:	None
Open Actions:	None
Viewers:	None
Features:	None
Layouts:	None

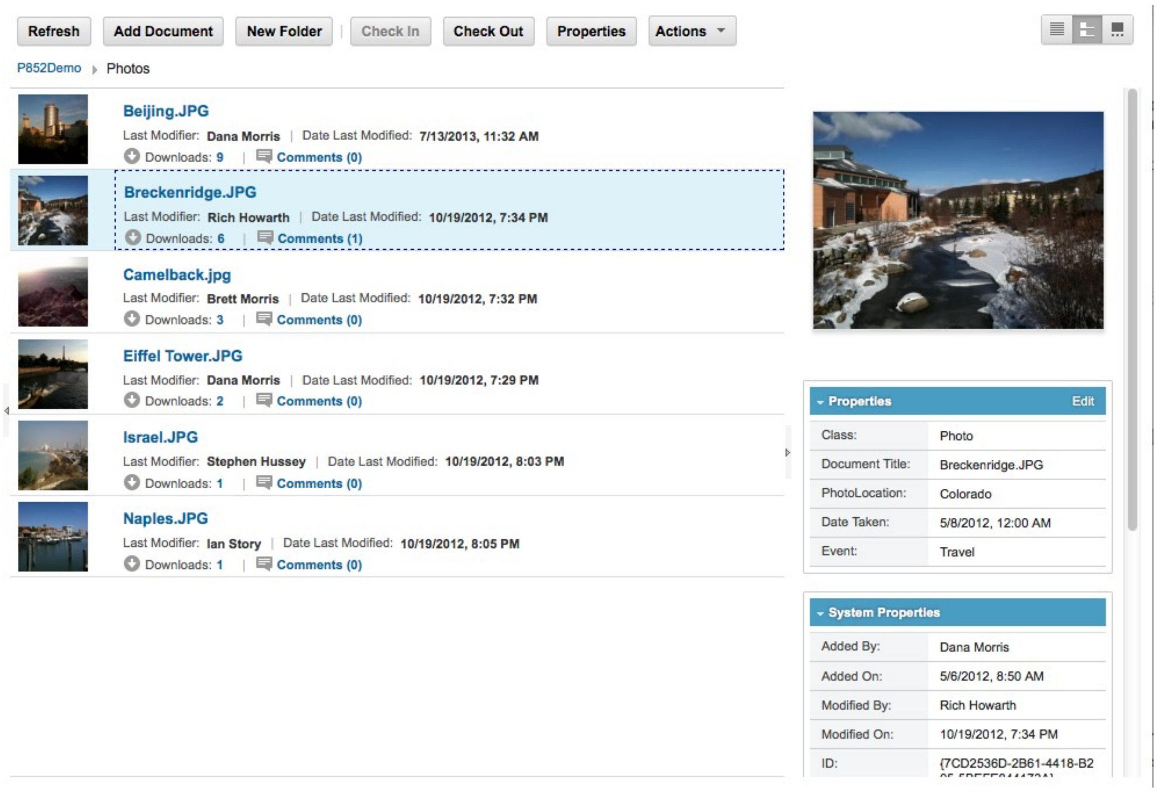
9/9/2013, 1:12 PM - The plug-in JAR file was not found at the specified URL. Ensure that JAR file is on the web application server and that the URL and file name are correct.

10. Click "**Save and Close**".

42.2 ContentList and Document information custom formatting

In the next step, we'll be importing various project files with the goal of enabling custom formatting of a property in both the ContentList and the Document information area. The ContentList is a DOJO Widget in the IBM Content Navigator toolkit, responsible for displaying lists of objects. It is used extensively throughout the IBM Content Navigator web application and provides a significant number of customization options. In this exercise we will demonstrate a simple customization that will format numbers in a property as currency.

The document information view is a module in the ContentList that is 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. In this exercise we will apply the "currency" formatting to the property in this view as well. Here is an example of the ContentList and Document information area in IBM Content Navigator:



Beijing.JPG
Last Modifier: Dana Morris | Date Last Modified: 7/13/2013, 11:32 AM
Downloads: 9 | Comments (0)

Breckenridge.JPG
Last Modifier: Rich Howarth | Date Last Modified: 10/19/2012, 7:34 PM
Downloads: 6 | Comments (1)

Camelback.jpg
Last Modifier: Brett Morris | Date Last Modified: 10/19/2012, 7:32 PM
Downloads: 3 | Comments (0)

Eiffel Tower.JPG
Last Modifier: Dana Morris | Date Last Modified: 10/19/2012, 7:29 PM
Downloads: 2 | Comments (0)

Israel.JPG
Last Modifier: Stephen Hussey | Date Last Modified: 10/19/2012, 8:03 PM
Downloads: 1 | Comments (0)

Naples.JPG
Last Modifier: Ian Story | Date Last Modified: 10/19/2012, 8:05 PM
Downloads: 1 | Comments (0)

Properties Edit

Class:	Photo
Document Title:	Breckenridge.JPG
PhotoLocation:	Colorado
Date Taken:	5/8/2012, 12:00 AM
Event:	Travel

System Properties

Added By:	Dana Morris
Added On:	5/6/2012, 8:50 AM
Modified By:	Rich Howarth
Modified On:	10/19/2012, 7:34 PM
ID:	{7CD2536D-2B61-4418-B2

The following steps will walk you through importing the sample files as well as describing what purpose each file / code snippet serves. At each step you should review the code to better understand how the plugin comes together.

Step 1 – The Plugin Java Files

The PluginResponseFilter Java files that will enable our custom formatters in the ContentList and Document Information area. A PluginResponseFilter is an extension that allows customization of the JSON response from the IBM Content Navigator server. In this case, we want to modify the structure of the ContentList to apply a "**decorator**" to the "**TotalCost**" property display. A decorator is a JavaScript method that alters the display of a property within the ContentList. In addition, our response filter will apply a custom "**propertyFormatter**" to the property displays in the document information area. The "propertyFormatter" is JavaScript method that formats the display of a value in the document information property grid.

The three Java files will enable the response filters for the search, open folder and open class actions in IBM Content Navigator.

The **esc1427Plugin.js** file is the base JavaScript file that will be loaded when IBM Content Navigator loads the plug-in. You can use this to JavaScript file to apply any global changes (such as a style override) or load any JavaScript classes that need to be available throughout the session. In this case, we're adding the JavaScript class to load our decorator JavaScript, which provides global methods that will be used within the ContentList to format our custom property.

The newly added files and the enhanced ESC1427Plugin.java file will enable two new response filters in the IBM Content Navigator application and these will insert additional entries in the JSON response from the server that will tell the IBM Content Navigator DOJO widgets to leverage the custom esc1427Decorator.js (for the ContentList) and the esc1427PropertyFormatter.js (for the Document information area) when displaying the "**TotalCost**" property.

Step 2 – Validating the plug-in

Next, we'll validate the plug-in is working.

Login to the admin desktop: http://localhost:9080/navigator?desktop=admin

1. Click on the "**Administration**" feature, which is the gear icon at the bottom of the navigation bar on the left side of the IBM Content Navigator User Interface. Credentials are filenetadmin/IBMFileNetP8

2. Click on "**Plugins**"

3. Select the "**IOD Session ESC-1427 Plug-in**" and select **Edit**

4. Click the "**Load**" button to make sure the plugin changes are reloaded

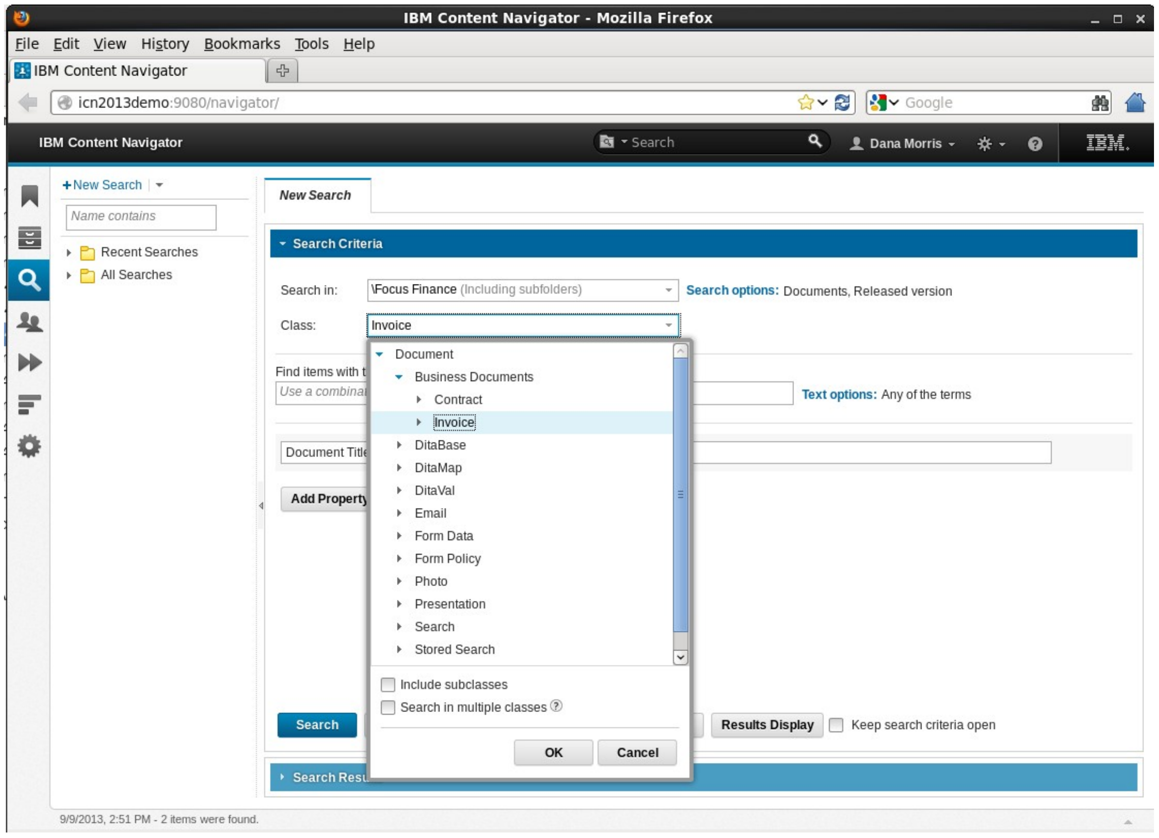
5. Log out of Content Navigator admin desktop and login to the ECM desktop

6. http://localhost:9080/navigator?desktop=ECM and click the browser refresh button to reload the page. Log back in using the user "**filenetadmin**" and password "**IBMFileNetP8**"

7. Select the "**Search**" feature by clicking on the magnifying glass icon in the navigation bar on the left side of the user interface.

8. Click on "**New Search**" located at the top of the search selector area.

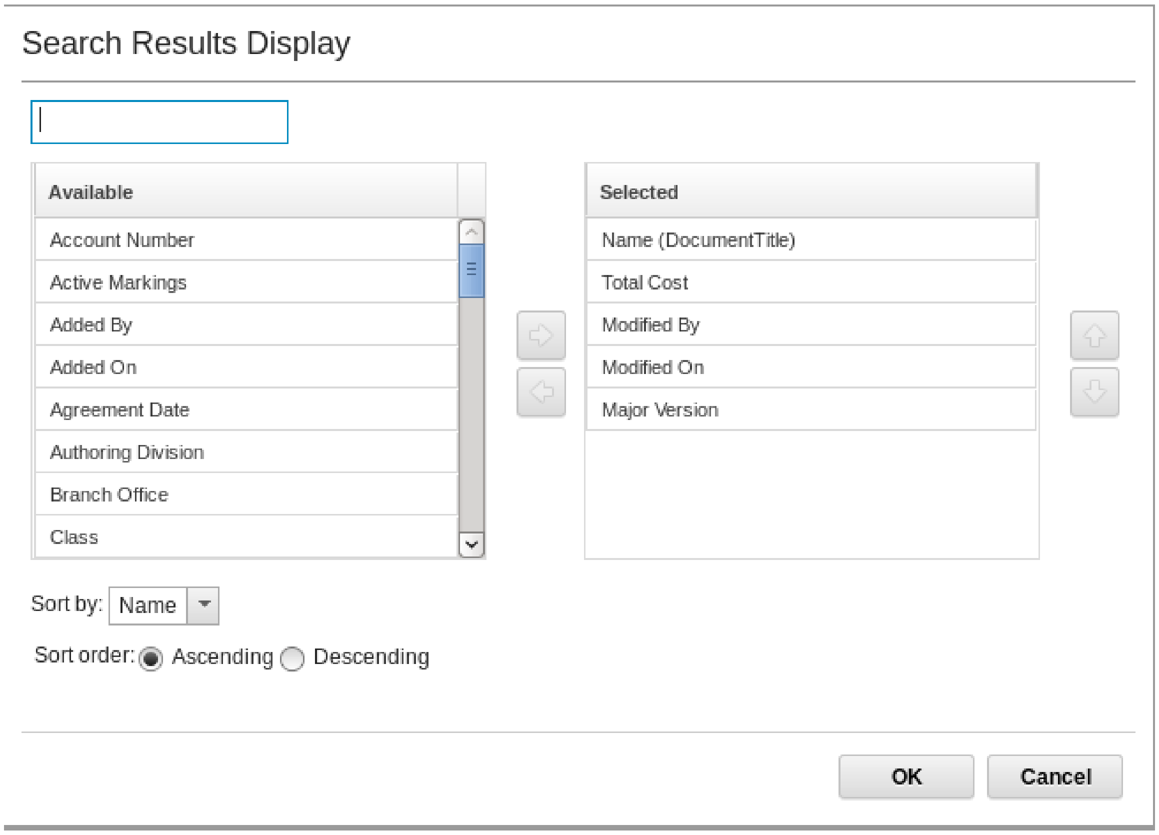
9. On the new search tab, open the class selector and navigate to the "**Invoice**" class located under Document ->



10. Select the **Invoice** class and select "**Is Not Empty**" as the search option for the Document Title property.

11. Click on the "**Results Display**" button.

12. Add the "**TotalCost**" property to the results display list:



Search Results Display

Available
Account Number
Active Markings
Added By
Added On
Agreement Date
Authoring Division
Branch Office
Class

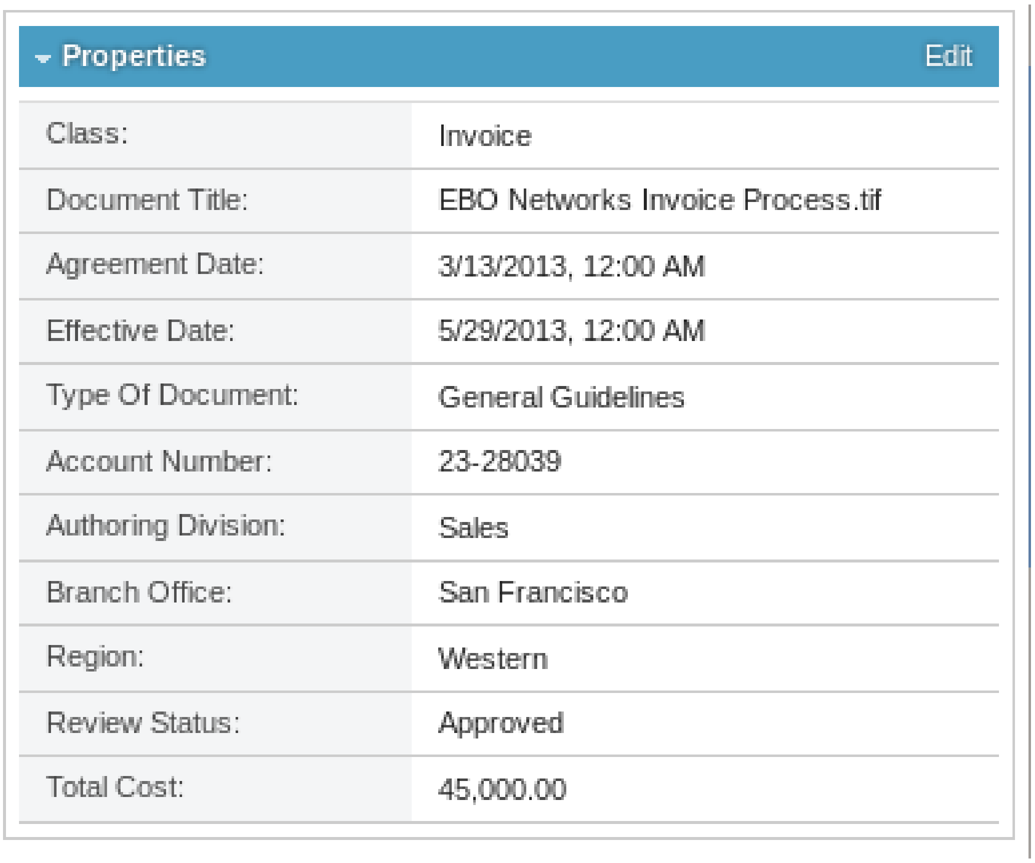
Selected
Name (DocumentTitle)
Total Cost
Modified By
Modified On
Major Version

Sort by: Name

Sort order: [● Ascending] ○ Descending

13. Click "**Search**"

In the search results you should see the Total Cost property is formatted as "currency" instead of a plain integer. If you select the document you will also see the value is formatting in the document information properties display:



Properties Edit

Class:	Invoice
Document Title:	EBO Networks Invoice Process.tif
Agreement Date:	3/13/2013, 12:00 AM
Effective Date:	5/29/2013, 12:00 AM
Type Of Document:	General Guidelines
Account Number:	23-28039
Authoring Division:	Sales
Branch Office:	San Francisco
Region:	Western
Review Status:	Approved
Total Cost:	45,000.00

42.3 Adding a custom property editor

In this step, we'll make further changes to our custom plugin to create a custom property editor for the "Photographer" property of the "Photo" class. Custom property editors can be used to control user input while adding and editing documents. This feature allows for much more granular control, by providing a mechanism for overriding the default IBM Content Navigator property editors. In this particular example, we will add a button next to the "Photographer" property that launches a user lookup dialog. The user will be forced to use this new dialog for finding and adding a user name to the "Photographer" attribute.

Step 1 – Import the Property Editor JavaScript class

1. Right-click the **com.ibm.ecm.iod.esc1427.WebContent.esc1427Dojo** folder in your project and then right click and select **Import** from the menu.

2. On the next screen, choose **File System**

3. Browse to the **"C:\Navigator\samplePlugins\esc1427Plugin_lab_6_3"** directory. Click OK and then check the box next to **esc1427PropertyEditor.js**.

   Additional Details: Importing this file will add the custom property editor DOJO widget to the plug-in. In this case, our custom property editor is an extension of the IBM Content Navigator ValidationTextBox DOJO widget, providing a button next to the standard input text area for a user lookup.

4. Click **Finish**.

Step 2 – Import the Property Editor template

1. Right-click the **com.ibm.ecm.iod.esc1427.WebContent.esc1427Dojo.templates** folder in your project and then right click and select **Import** from the menu.

2. On the next screen, choose **File System**

3. Browse to the **"C:\Navigator\samplePlugins\esc1427Plugin_lab_6_3"** directory. Click OK and then check the box next to **ESC1427PropertyEditor.html**.

   Additional Details: A template is an HTML fragment that provides the base HTML layout for a DOJO widget. In this case, our template adds the button and the style classes necessary to place the button next to the input box in the browser.

4. Click **Finish**.

Step 3 – Import the new plug-in Style Sheet

1. Right-click the **com.ibm.ecm.iod.esc1427.WebContent** folder in your project and then right click and select **Import** from the menu.

5. On the next screen, choose **File System**

6. Browse to the **"C:\Navigator\samplePlugins\esc1427Plugin_lab_6_3"** directory. Click OK and then check the box next to **esc1427Plugin.css**.

   Additional Details: Here we're adding the a set of styles necessary to position the button next to the text input in the browser.

7. Click **Finish**.

Step 4 – Overwrite the Plugin and OpenClass response filter Java class

In this step we'll also overwrite the existing Plugin Java class as well as our OpenClass response filter Java class. The new versions will enable the newly added style sheet as well as the additional logic to update the OpenClass JSON response to ensure the new custom property editor is enabled for the "Photographer" property.

1. Right-click the **com.ibm.ecm.iod.esc1427** folder in your project and then right click and select **Import** from the menu.

2. On the next screen, choose **File System**

3. Browse to the **"C:\Navigator\samplePlugins\esc1427Plugin_lab_6_3"** directory. Click OK and then check the box next to **ESC1427Plugin.java** and **ESC1427OpenClassResponseFilter.java**.

4. Click **Finish.**

Step 5 – Validate the plug-in

In this step we'll validate the plug-in is enabled by adding a new Photo and leveraging using the new custom property editor to select the photographer.

1. Open Firefox and to IBM Content Navigator again (ECM desktop) using user "**filenetadmin**" and password "**IBMFileNetP8**".

2. Click on the "**Administration**" feature, which is the gear icon at the bottom of the navigation bar on the left side of the IBM Content Navigator User Interface.

3. Click on "**Plugins**"

4. Select the "**IOD Session ESC-1427 Plug-in**" and select **Edit**

5. Click the **Load** button to make sure the plugin changes are reloaded and click **Save and Close**

6. Log out of Content Navigator and click the browser refresh button to reload the page. Log back in using the user "**filenetadmin**" and password "**IBMFileNetP8**"

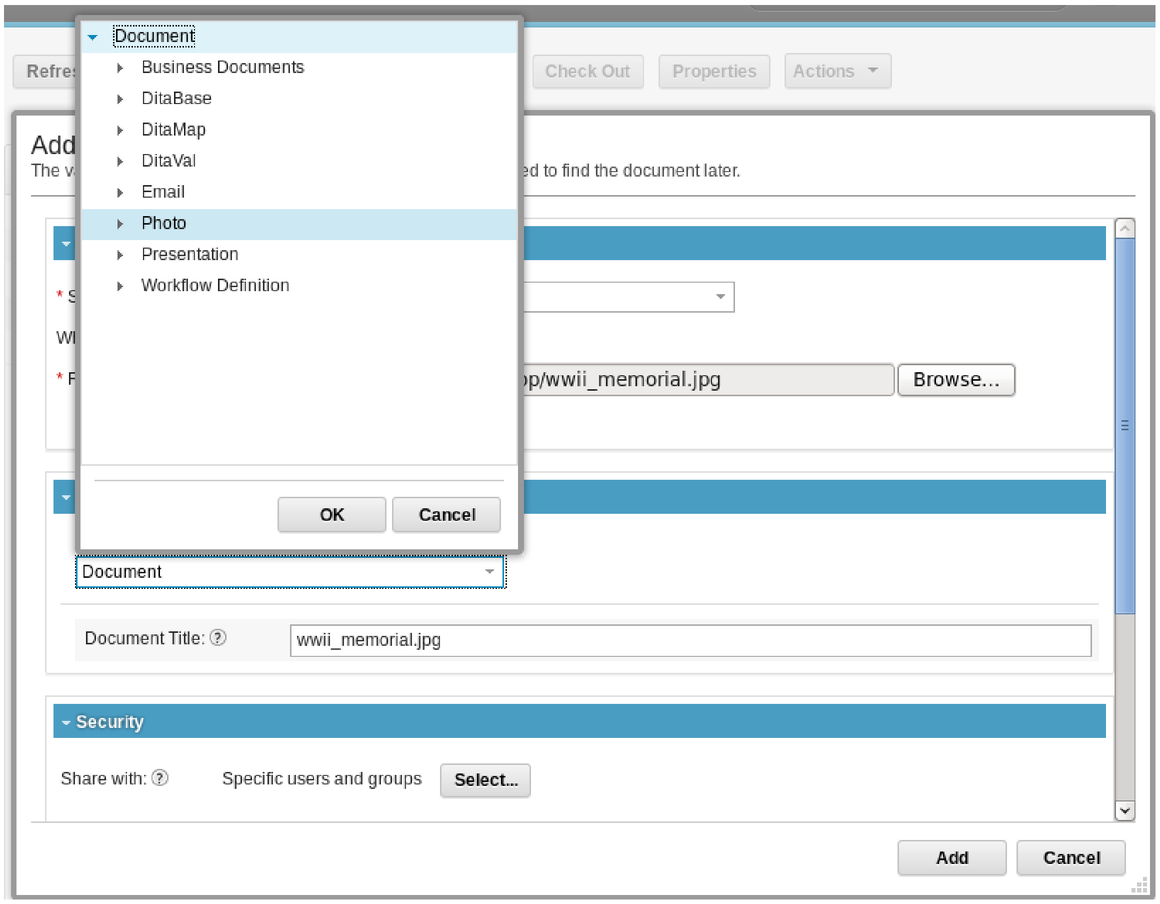
7. Select the **Browse** feature by clicking on the second icon in the navigation bar on the left side of the user interface.

8. Click on the folder named "**Photos**"

9. Click "**Add Document**"

10. Click the **Browse** button and select the file named "**WWII-Memorial020.jpg**" under C:\Users\Administrator\Downloads folder:

11. Click the "**Class**" selector and select the class "**Photo**".



After selecting the class you should see a button called “Lookup” next to the “Photographer” property. You can now click this button to select a user and add the document!

42.4 Creating the “Windows Explorer” style browse

In this step, we’ll make the final changes to our custom plugin to create a new browse feature that will provide users with a “windows explorer-like” view into the Enterprise Content Management repository. To accomplish this we will extend the existing IBM Content Navigator browse feature and add the file menu bar, similar to what you would see in a Windows program. Next, we will alter the default menus to remove actions relating to features not available in Browse. Finally we will create a new desktop that will enable our new feature and it’s menus.

Step 1 – Import the Java classes and overwrite the existing ESC1427Plugin Java class

In this step we will import the new classes that will define our custom Browse feature as well as the customized versions of the existing IBM Content Navigator “Item” menus.

1. Right-click the **com.ibm.ecm.iod.esc1427** folder in your project and then right click and select **Import** from the menu.

2. On the next screen, choose **File System**

3. Browse to the **"C:\Navigator\samplePlugins\esc1427Plugin_lab_6_4"** directory. Click OK and then check the box next to **ESC1427Plugin.java, ESC1427Feature.java, ESC1427FolderContextMenu.java, ESC1427ItemContextMenu.java, ESC1427MixItemsContextMenu.java and ESC1427SystemItemContextMenu.java**.

   Additional Details: Plug-ins have the ability to define custom menu types and toolbar types as well as the ability to define menu and toolbars. In this case, we're providing custom menus of existing IBM Content Navigator menu types. These custom menus remove some actions that will not be available in this simplified view of the repository.

4. Click **Finish.**

Step 2 – Overwrite the existing Style Sheet

In this step we will add new styles to the plugin style sheet to customize the look and feel of the browse panel.

1. Right-click the **com.ibm.ecm.iod.esc1427.WebContent** folder in your project and then right click and select **Import** from the menu.

2. On the next screen, choose **File System**

3. Browse to the **"C:\Navigator\samplePlugins\esc1427Plugin_lab_6_4"** directory. Click OK and then check the box next to **esc1427Plugin.css**.

4. Click **Finish.**

Step 3 – Add the new Browse widget

18. Right-click the **com.ibm.ecm.iod.esc1427.WebContent.esc1427Dojo** folder in your project and then right click and select **Import** from the menu.

19. On the next screen, choose **File System**

20. Browse to the **"C:\Navigator\samplePlugins\esc1427Plugin_lab_6_4"** directory. Click OK and then check the box next to **ESC1427BrowsePane.js**.

    Additional Details: The ESC1427BrowsePane is an extension on the existing IBM Content Navigator BrowsePane. The extension is necessary to add the menubar (similar to what you see in a Windows program) and alter the custom modules applied to the ContentList widget.

21. Click **Finish.**

Step 4 – Add the new Browse html template

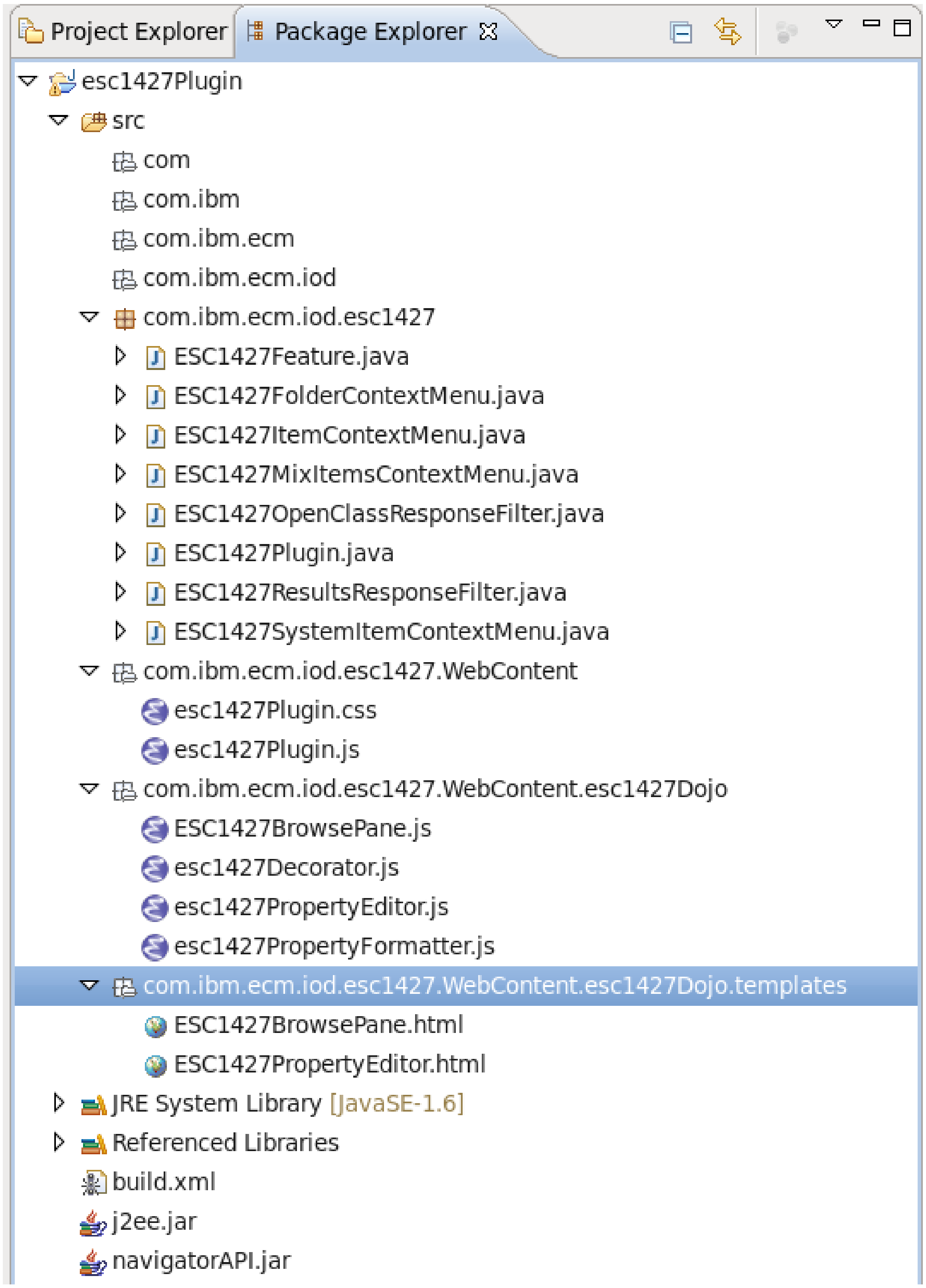
1. Right-click the **com.ibm.ecm.iod.esc1427.WebContent.esc1427Dojo.templates** folder in your project and then right click and select **Import** from the menu.

2. On the next screen, choose **File System**

3. Browse to the **"C:\Navigator\samplePlugins\esc1427Plugin_lab_6_4"** directory. Click OK and then check the box next to **ESC1427BrowsePane.html**.

4. Click **Finish.**

   After completing the first four steps, your eclipse project should look like this:



```
esc1427Plugin
  src
    com
    com.ibm
    com.ibm.ecm
    com.ibm.ecm.iod
    com.ibm.ecm.iod.esc1427
      ESC1427Feature.java
      ESC1427FolderContextMenu.java
      ESC1427ItemContextMenu.java
      ESC1427MixItemsContextMenu.java
      ESC1427OpenClassResponseFilter.java
      ESC1427Plugin.java
      ESC1427ResultsResponseFilter.java
      ESC1427SystemItemContextMenu.java
    com.ibm.ecm.iod.esc1427.WebContent
      esc1427Plugin.css
      esc1427Plugin.js
    com.ibm.ecm.iod.esc1427.WebContent.esc1427Dojo
      ESC1427BrowsePane.js
      esc1427Decorator.js
      esc1427PropertyEditor.js
      esc1427PropertyFormatter.js
    com.ibm.ecm.iod.esc1427.WebContent.esc1427Dojo.templates
      ESC1427BrowsePane.html
      ESC1427PropertyEditor.html
  JRE System Library [JavaSE-1.6]
  Referenced Libraries
  build.xml
  j2ee.jar
  navigatorAPI.jar
```

Step 4 – Setup the new desktop to enable the custom Browse feature

1. Open Firefox and to IBM Content Navigator again (http://localhost:9080/navigator?desktop=ECM) using user "**filenetadmin**" and password "**IBMFileNetP8**" and access the "**Administration**" feature.

2. Click on "**Plugins**"

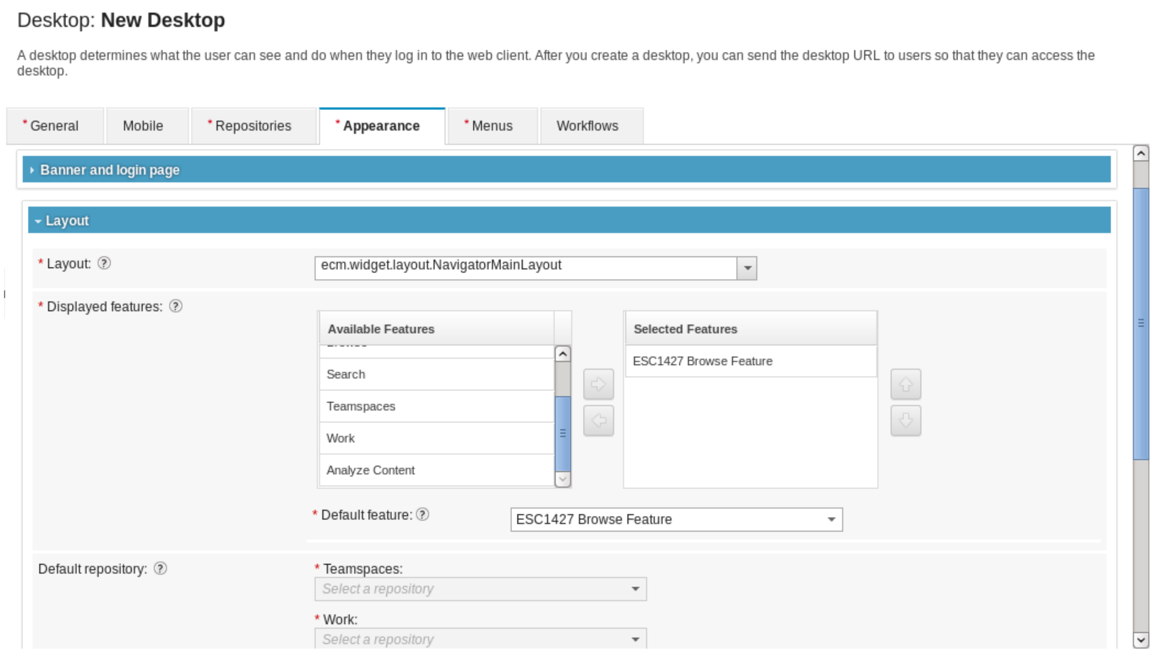
3. Select the "**IOD Session ESC-1427 Plug-in**" and select **Edit**

4. Click the **Load** button to make sure the plugin changes are reloaded and click **Save and Close**

5. Next, click on **Desktops** and click the **New Desktop** button. On the general tab, enter **esc1427** as the name.

6. Under the "**Authentication**" section on the "**General**" tab, select the only repository available "**ECM**".

7. On the **Appearance** tab, enter **ESC1427** as the **Application Name** then collapse the **Banner and Login Page** section. In the **Selected Features** box, select all of the features and click the left arrow to remove them from the desktop. In the **Available Features** box, select the **ESC1427 Browse Feature** and click the right arrow to add it to the desktop.


Desktop: New Desktop

A desktop determines what the user can see and do when they log in to the web client. After you create a desktop, you can send the desktop URL to users so that they can access the desktop.

* Layout: ecm.widget.layout.NavigatorMainLayout
* Displayed features:

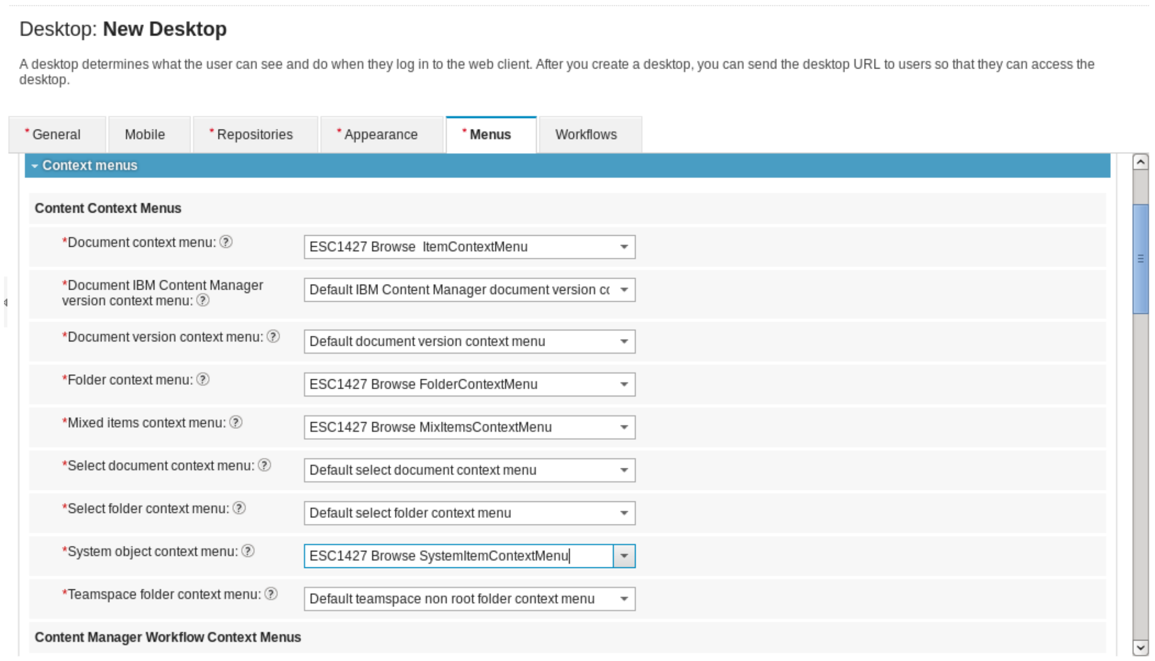
Available Features	Selected Features
Search	ESC1427 Browse Feature
Teamspaces	
Work	
Analyze Content	

* Default feature: ESC1427 Browse Feature

Default repository:
* Teamspaces: Select a repository
* Work: Select a repository

8. Click the **Menus** tab. Collapse the **Toolbars** section.

9. Update the **Document context menu**, **Folder context menu**, **Mixed items context menu**, and the **System object context menu** so that each definition is using the **ESC1427** menu definition instead of the default entry. When you're done the desktop definition should look something like the following picture.


Desktop: **New Desktop**

A desktop determines what the user can see and do when they log in to the web client. After you create a desktop, you can send the desktop URL to users so that they can access the desktop.

*General | Mobile | *Repositories | *Appearance | ***Menus** | Workflows

Context menus

Content Context Menus

*Document context menu:	ESC1427 Browse ItemContextMenu
*Document IBM Content Manager version context menu:	Default IBM Content Manager document version c
*Document version context menu:	Default document version context menu
*Folder context menu:	ESC1427 Browse FolderContextMenu
*Mixed items context menu:	ESC1427 Browse MixItemsContextMenu
*Select document context menu:	Default select document context menu
*Select folder context menu:	Default select folder context menu
*System object context menu:	ESC1427 Browse SystemItemContextMenu
*Teamspace folder context menu:	Default teamspace non root folder context menu

Content Manager Workflow Context Menus

1. Click **Save and Close**. Log out and close the browser.

2. Reopen Mozilla Firefox. Enter http://base-win2k8x64:9080/navigator/?desktop=**esc1427** to load your custom desktop.

Lab Activity 7 – Building a custom repository search service

This chapter describes how to create an IBM Content Navigator plug-in to extend the base functionality of the product. We introduce how to create a custom repository search service and show the results in existing ContentList widgets. With the custom repository search service, customers can create their own tailored search experience while still being able to view the results using the existing Content Navigator ContentList widget.

This chapter covers the following topics:

- Overview of the example
- Viewing results in the ContentList widget
- Custom repository search service in the sample plug-in
- Sample query strings
- Creating a new plug-in with a custom repository search service
- Adding new capabilities to the existing search service

1. *Example Overview*

IBM Content Navigator includes a SearchTemplate JavaScript class as the repository search model. If you want to extend IBM Content Navigator and add new functionality then you can use the SearchTemplate model to build your own search templates.

There are many reasons why you may want to build your own custom search templates. One reason could be that the current search template builder does not support complex Boolean operations such as combining AND / OR / NOT. Another possible reason could be that you have a requirement to execute searches against a separate database but still leverage the advanced search results view provided by the ContentList widget of IBM Content Navigator. In these situations, a custom repository search service will be needed in order to provide the functionality to your end-users.

The example shown in this chapter is an IBM Content Navigator plug-in that provides a custom repository search service and displays the results in the existing ContentList widget.

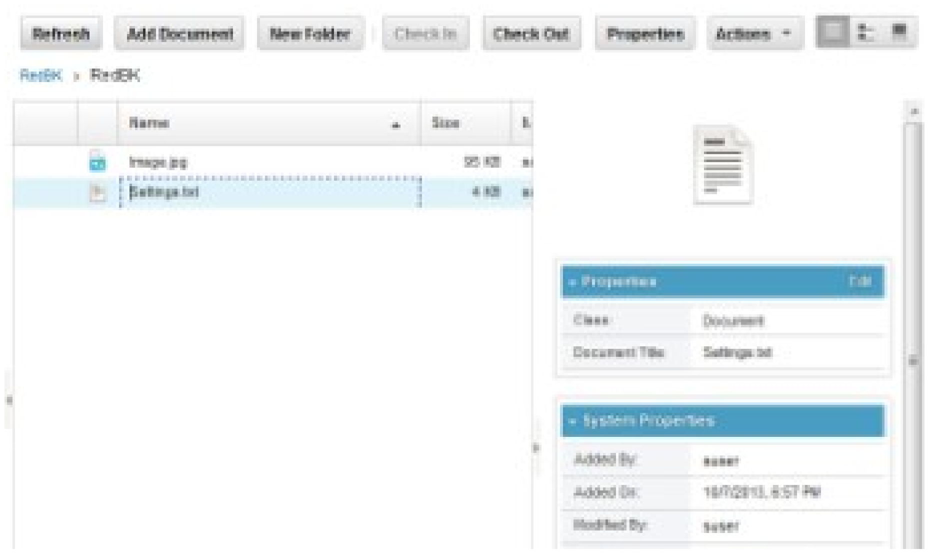
2. *Viewing results in the ContentList widget*

When you use the custom repository search service, you need to consider how to display the search results to your end-users. IBM Content Navigator provides a powerful widget called ContentList that is used to display and manipulate lists of items throughout the Content Navigator UI. The ContentList is a widget that supports pluggable modules for adding various types of functionality. The modular approach allows the programmer to easily add and remove a new module or to enable or disable an existing module. The ContentList is easily one of the most powerful widgets in Content Navigator and you are encouraged to make use of this widget wherever and whenever possible.

At the most basic level, the ContentList widget implements a spreadsheet style grid structure. The ContentList widget is a wrapper around the Gridx widget provided by the IBM Dojo Extension Toolkit (IDX). ContentList adds many enhancements and customizable modules to the base Gridx component. For example, while Gridx provides a base grid view that is exposed in the ContentList as the ViewDetail module, there are two additional views that ContentList adds called ViewMagazine and ViewFilmStrip. These views provide very different and unique ways of working with content while still sharing the same base features.

Below is a screenshot of the out-of-the-box ContentList in Content Navigator that has all three views enabled. The user can switch between views using the three square icons on the top right of the ContentList. Also notice that there is a toolbar module that shows at the top of the screenshot. The toolbar contains several actions the user can take against selected content in the ContentList. Below the toolbar module, there is a breadcrumb module that shows the current path. On the right hand side there is the document information module that displays the currently selected document's class property values and thumbnail image.

Detail view of ContentList



Name	Size
Image.jpg	55 KB
Settings.txt	4 KB

Class	Document
Document Title	Settings.txt
Added By:	suser
Added On:	10/7/2013, 6:57 PM
Modified By:	suser

ContentList Modules

Here is the full list of ContentList modules that are provided out-of-the-box by IBM Content Navigator:

- ContentList: the core module that displays the data and the launch actions.
- Toolbar: provides a toolbar containing actions.
- Bar: controls the arrangement of the widgets displayed in the top bar of the ContentList.
- Breadcrumb: displays the currently select folder path.
- DocInfo: detailed class specific property values, system property values, and thumbnail image for the currently selected content item.

- FilterData: adds the filter box to quickly filter to a particular item in the list.

- InlineMessage: inline message display.

- TotalCount: displays the total result count for queries run against FileNet Content Platform
E Engine 5.2. This will be hidden for repositories that do not provide the total count information.

- ViewDetail: the detail or list view.

- ViewMagazine: magazine view.

- ViewFilmStrip: filmstrip view.

- RowContextMenu: enables right-click context menus for the current row.

- DndFromDesktopAddDoc: allows users to drag-and-drop from their desktop in to the grid (FireFox, Chrome, Safari, IE10+ only)

- DndRowMoveCopy: provides the ability to drag-and-drop a row, moving the content.

- DndRowCopy: provides the ability to drag-and-drop a row, copying the content to the new location.

- DndDropOnly: a grid module that extends DndRowMoveCopy. It can disable the dragging of
r rows.

Below is an excerpt of code from the SampleFeaturePane.js file in the Content Navigator sample plug-in. This example code shows how to set the modules for the ContentList widget. You can create your own context menu module and replace the system one. There is a setting in the Content Navigator desktop configuration to control if the ViewFilmStrip view is shown for the desktop - the sample code here honors this desktop configuration setting.

```
getContentListGridModules: function() {

        var array = [];
        array.push(DndRowMoveCopy);
        array.push(DndFromDesktopAddDoc);
        array.push(RowContextMenu);
        return array;
   }    },
       getContentListModules: function() {
        var viewModules = [];
        viewModules.push(ViewDetail);
        viewModules.push(ViewMagazine);
        if (ecm.model.desktop.showViewFilmstrip) {
          viewModules.push(ViewFilmStrip);
        }
        var array = [];
        array.push(DocInfo);

array.push({ moduleClass: Bar,
          top: [[[], { "className": "BarViewModules"}] ]] });
        return array; },
```

To load the actual results set in to the ContentList widget, use the setResultSet (model, list Parent) method. If the result set comes from the IBM Content Navigator model, like the SearchTemplate search result, then the resultSet can be set directly. If the result set comes from a custom search request, such as one which you've directed at a database, then the result set needs to be constructed according to the ResultSet model format. This format can be constructed by using IBM Content Navigator's JSON APIs. These APIs provide class definitions for a result set as well as classes for defining columns and rows for display.

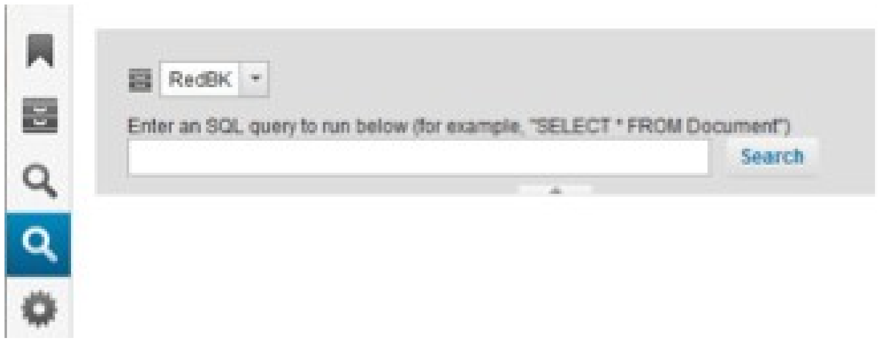
Helpful links

- Basic Gridx Usage http://oria.github.io/gridx/
- ContentList package JSDoc in the IBM Content Navigator Information Center: http://pic.dhe.ibm.com/infocenter/cmgmt/v8r4m0/topic/com.ibm.developingeuc.doc/eucrf015.htm

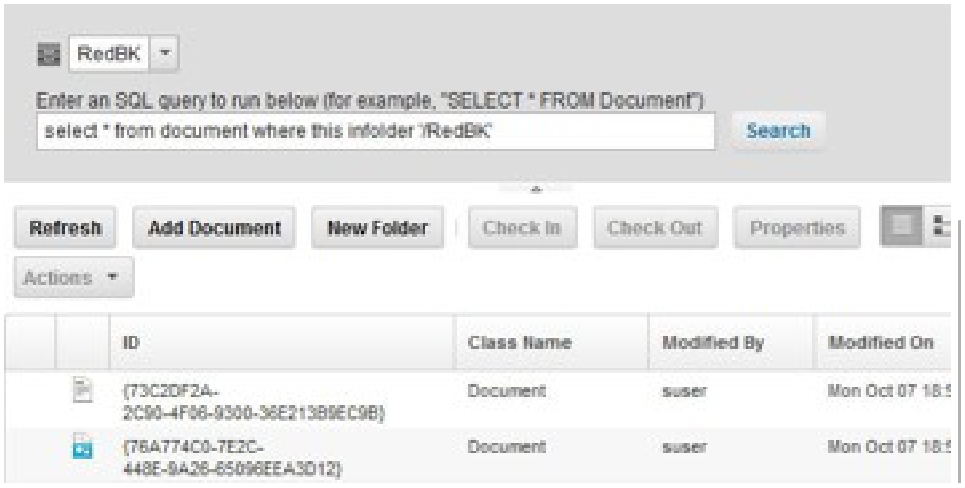
3. Custom repository search service in the sample plug-in

• IBM Content Navigator provides a sample repository search service in the sample plug-in. Before implementing the custom repository search service, it's useful to first try it out and see how it works. To see how it works you need to register the sample plug-in to your IBM Content Navigator deployment and then add the Sample feature to the desktop and save the changes. For this lab, we've already done this step for you so you can jump straight to testing the plugin.

After you add the sample feature to the desktop, you can see it listed in the feature pane of the desktop (see below). Click on the sample feature icon to see a search bar that lets you search in the repository directly using SQL style queries.


RedBK
Enter an SQL query to run below (for example, "SELECT * FROM Document")
Search

Enter a valid query string and click the **Search** button. The results of the search request will be shown in the ContentList widget below the search bar.



Enter an SQL query to run below (for example, "SELECT * FROM Document")

select * from document where this infolder '/RedBK'

ID	Class Name	Modified By	Modified On
{73C2DF2A-2C90-4F06-9300-36E213B9EC9B}	Document	suser	Mon Oct 07 18:5
{76A774C0-7E2C-448E-9A26-65096EEA3D12}	Document	suser	Mon Oct 07 18:5

4. *Sample code for the custom repository search service*

Content Navigator provides the sample feature shown above as an out-of-the-box sample. The samples are located in the Content Navigator installation directory, typically , C:\Program Files (x86)\IBM\ECMClient\samples subdirectory.

In the sample plug-in project, there are three Java files related to the custom repository search service.

- SamplePluginSearchService.java
- SamplePluginSearchServiceCM.java
- SamplePluginSearchServiceP8.java

SamplePluginSearchService.java is the core service that extends the PluginService class. There are three methods in this file:

getId()
execute()
writeResponse()

The getId() method returns the service id that will be used in the JavaScript code.

The execute() method implements the primary functionality of the service. It first gets the repository information from the request, followed by the query string the user entered in to the search bar. The callbacks.getSynchObject() method tries to retrieve the synchronized repository object. If there is one already present, then the code will synchronize around that object and then execute the search. If a synchronized repository object is not present, then the method will simply execute the search directly. Finally, the execute method will write the results of the request in to the response to send back to the client.

NOTE: The getSynchObject() method is only implemented for IBM Content Manger or IBM Content Manager OnDemand repositories. For IBM FileNet Content Manager repository types, this method will always return null.

The writeResponse() method writes the response back just as its name indicates. However, you should note the response fomat, which includes a JSON prefix as well as an attempt to zip the JSON response for browsers and clients that support the gzip format.

SamplePluginSearchServiceCM and SamplePluginSearchServiceP8 provide the repository specific implementation details that execute the search request. These classes both build the response and are invoked by the SamplePluginSearchService class. When these classes are invoked, they perform the following operations:

1. Build a basic resultSet object

2. Execute the query request

3. Get privilege masks for each result item

4. Add attributes to the resultSet

The most important step is building the resultSet structure. In the buildP8ResultStructure() method, you can see how to add columns to the resultSet, how to add columns specifically for magazine view, and more.

The SampleFeaturePane.js runSearch() method invokes the samplePluginSearchService when users click on the **Search** button. The button click event and Dojo methods are connected in the SampleFeaturePane.html template. The resultSet is built in the Java service, so it can be set directly in to the ContentList widget with a name of 'searchResults'. The searchResults object is defined in the SampleFeaturePane.html template.

5. Creating a new plug-in with a custom repository search service

If you want to create a new plug-in to contain the custom repository search, the easiest way is to use the existing code in the sample plug-in whenever possible. In this section, we will show you how to add the existing custom repository search service into a custom plug-in.

Step 1 – Create a new plug-in project

In case you have closed Eclipse, you'll need to restart the client for the exercise. In order to do that, click on Applications->System Tools and select File Browser to launch the File Browser. On the left hand tree view, select the eclipse folder and double click on the eclipse icon to launch Eclipse.

In Eclipse, choose **File->New->Other**. In the dialog box, expand the IBM Content Navigator folder and select the Content Navigator Plug-in project type. Follow the prompts to create a new plug-in named **CustomSearchPlugin** with a package name of **com.ibm.ecm.extension.customsearch**. When prompted for the navigatorAPI.jar file, browse to C:\Program Files (x86)\IBM\ECMClient\lib to find the JAR file.

Step 2 – Import the search service from the sample plug-in

After you have created the plug-in project, create the search service in the CustomSearchPlugin project by following these steps:

1. Expand the **CustomSearchPlugin** project in Eclipse and expand the **src** node.

2. Right click on the package **com.ibm.ecm.extension.customsearch** and select **IBM Content Navigator ✹ Server Extensions ✹ New Service....**

3. In the class name text box enter a value of **SearchService** and click OK.

4. Right click on the project and select Properties. Select Java Build Path and then select the option to Add External JARs…

    a. Add the Jace.jar file in to the build path of the CustomSearchPlugin project. You can find this JAR file in the C:\Program Files (x86)\IBM\ECMClient\configure\explodedformat\navigator\WEB-INF\lib directory

    b. Add the struts-1.1.jar.

    c. Add the commons-codec-1.4.jar

5. Copy the SamplePluginSearchServiceP8 Java file into the **com.ibm.ecm.extension.customsearch** package. You can find this file in the **C:\Program Files (x86)\IBM\ECMClient\samples\samplePlugin\src\com\ibm\ecm\extension\sample** directory. After doing so you will see compilation errors with the class. You will need to update the package definition to be "c**om.ibm.ecm.extension.customsearch**".

6. Add the invoke logic into the SearchService.java. You can do so by copying the code from the Content Navigator SamplePluginSearchService class as follows. The SamplePluginSearchService Java source file should be located in the **C:\Program Files (x86)\IBM\ECMClient\samples\samplePlugin\src\com\ibm\ecm\extension\sample** directory:

– Copy the import statements over to SearchService.java

– Copy the definitions of REPOSITORY_ID, REPOSITORY_TYPE and QUERY to SearchService.java.

– Copy the execute() method and writeResponse() method from the SamplePluginSearchService.java file in to the SearchService.java class in your project . Make sure
t to remove the old empty execute() method from SearchService.java. There may be some import
e errors pointing to the sample plug-in path of the following two files. Remove the two import lines. SamplePluginSearchServiceCM.java SamplePluginSearchServiceP8.java

– Remove any references to SamplePluginSearchServiceCM by removing the IF statements for "if CM". Eclipse will mark these items with red to indicate there are compilation errors so they will be easy to spot.

NOTE: Before you can use the Ant build file (build.xml) to compile the project, you need to first update the build.xml to include the full paths to the Jace.jar and struts-1.1.jar files.

Now your CustomSearchPlugin project has the custom repository search service from the sample plug-in.

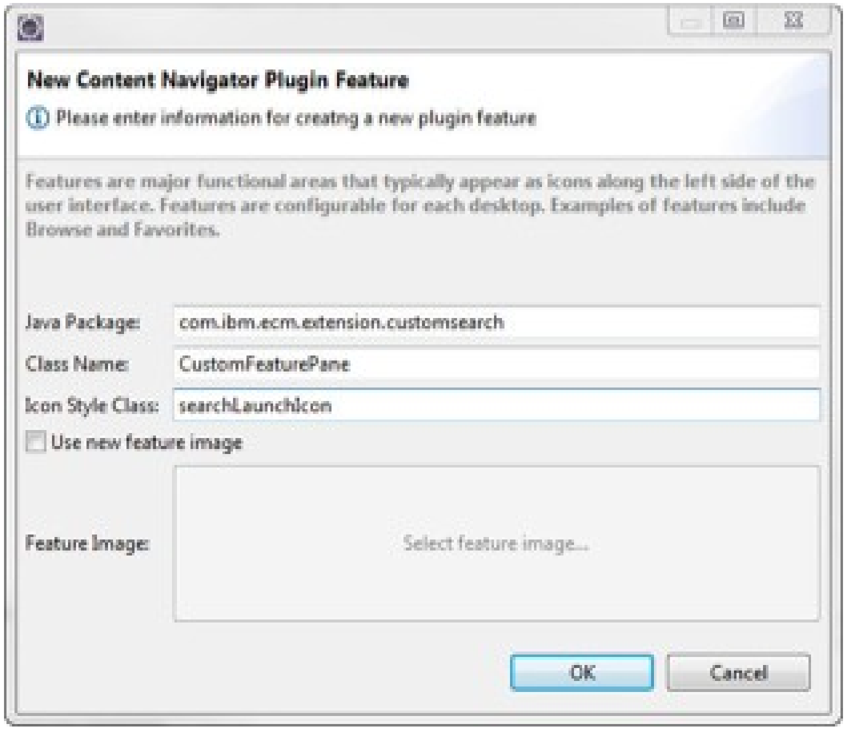
Step 3 – Import the feature pane from the sample plug-in

Let's create a new feature now in CustomSearchPlugin project:

1. Expand the project in Eclipse and expand the **src** node.

2. Right click on the package **com.ibm.ecm.extension.customsearch**.

3. Select the **IBM Content Navigator✹ New Feature…**. option. Enter a class name of **CustomFeaturePane**.



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.extension.customsearch

Class Name: CustomFeaturePane

Icon Style Class: searchLaunchIcon

Use new feature image

Feature Image: Select feature image...

In the dialog, the Icon Style Class is the CSS class that will display an icon to the user to represent this new feature in the Content Navigator feature bar. We will re-use the existing **searchLaunchIcon** class that is provided by Content Navigator and is used by the SamplePluginFeature class. You course of course choose to build your own class that leverages a custom icon but to keep this exercise simple we'll just reuse an existing one.

After clicking OK, the Content Navigator Eclipse Plugin will create a new, empty feature class. At this point you could choose to test the new feature by compiling it and registering the plugin in to IBM Content Navigator.

Now take a closer look at the CustomSearchPlugin project and note what actually took place when you clicked the OK button to create the feature. Unlike extending an existing service, there are several files that are created for our feature:

- CustomFeaturePane.java
- CustomFeaturePane.js
- CustomFeaturePane.html

A new feature usually has its own Dojo pane and a corresponding HTML template file. Now you can copy the feature related code from the sample plug-in in to your new empty feature:

- For the CustomFeaturePane.html file, replace its contents with the contents of the SamplePluginFeaturePane.html file in the Content Navigator sample plug-in.

- This file is located in the **C:\Program Files (x86)\IBM\ECMClient\samples\samplePlugin\src\com\ibm\ecm\extension\sample\WebContent\samplePluginDojo\templates** directory.

- For the CustomFeaturePane.js file, replace its contents with the contents of the SamplePluginFeaturePane.js file. This file is located in the **C:\Program Files (x86)\IBM\ECMClient\samples\samplePlugin\src\com\ibm\ecm\extension\sample\WebContent\samplePluginDojo\** directory.

- Make additional adjustments to the CustomFeaturePane.js file as follows.

  - The last line of the define section needs to be modified to: **dojo/text!./templates/CustomFeaturePane.html**. This change is needed because the template we are using is named CustomFeaturePane.html in this project, but the line in SamplePluginFeaturePane.js points to a different template file from the Content Navigator sample plugin.

  - The first value of the return declare line needs to be changed from *samplePluginDojo.SampleFeaturePane* to *customSearchPluginDojo.CustomFeaturePane*

  - Find the **Request.invokePluginService** line in the runSearch() method and change it to the following text: Request.invokePluginService("CustomSearchPlugin", "SearchService", The first value is the plug-in ID defined in the CustomSearchPlugin.getId() method. The second value is the search service ID defined in SearchService.getId(). This is how the service is invoked.

Step 4 – Build and test the new plug-in

It is now time to build and deploy the new plug-in so we can test it in a running Content Navigator instance.

1. Find the build.xml in your project, right click on it and select Open.

2. Add the following entries to the classpath section:

```
<pathelement location="C:/Program Files
(x86)/IBM/ECMClient/configure/explodedformat/navigator/WEB-INF/lib/Jace.jar"
/>
<pathelement location="C:/Program Files
(x86)/IBM/ECMClient/configure/explodedformat/navigator/WEB-INF/lib/struts-
1.1.jar" />
<pathelement location="C:/Program Files
(x86)/IBM/ECMClient/configure/explodedformat/navigator/WEB-INF/lib/commons-
codec-1.4.jar" />
```

3. Save the changes to the build.xml file.

4. Right click on the build.xml file in the project explorer and select **Run As ✹ Ant Build**.

5. Open the IBM Content Navigator admin desktop using the following URL:

   http://base-win2k8x64*:9080/navigator?desktop=admin*

6. Log in as filenetadmin / IBMFileNetP8

7. Click on the **Plug-ins** item. Click the **New Plug-in** button. Select the "**Class file path**" option. For the **Class file path**, enter C:\Users\Administrator\workspace\CustomSearchPlugin\bin. For the **Class name**, enter **com.ibm.ecm.extension.customsearch.CustomSearchPlugin** and then click **Load**. The information of the plug-in appears.

   NOTE: Any time that you recompile your code, you should return to the plugins view and reload the plugin file.

8. Click Save and Close.

9. Open the Demo desktop definition and click on the **Appearance** tab. Select the **CustomFeaturePane** and add it to the Selected Features.

10. Click Save and Close.

11. Logout of Content Navigator.

12. Test the new feature by accessing IBM Content Navigator again in the browser:

    http://base-win2k8x64*:9080/navigator*

If you can see the sample plug-in feature pane showing up twice in the feature bar, then it is successful.

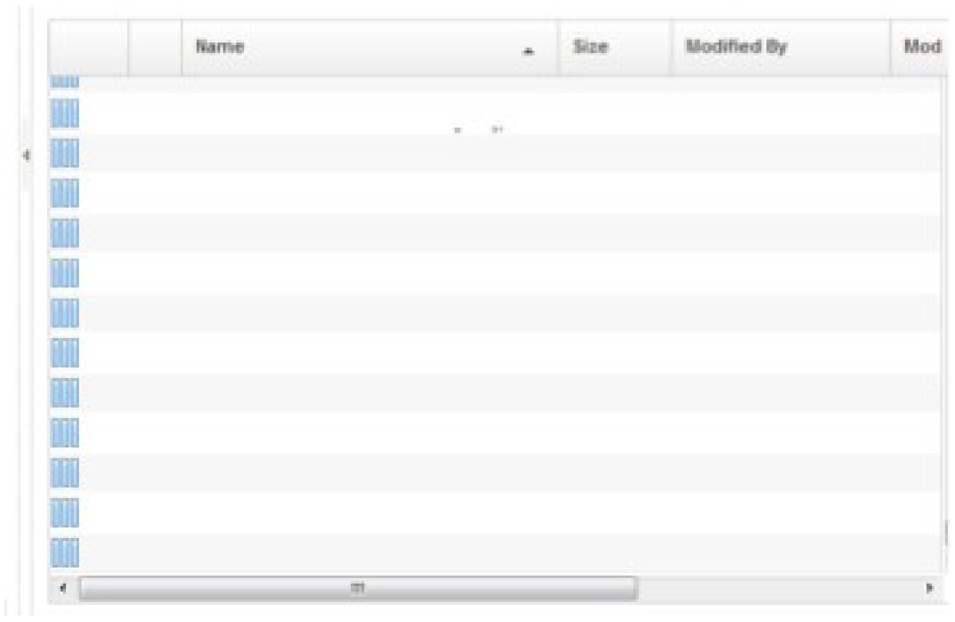
6. Add a new function to the existing search service

Now that we have a custom repository search service pulled over from the sample plug-in, we'll add some custom logic to the new search service to address our sample use-case.

Step 1 – Add a paging function

In the SamplePluginSearchServiceP8 class, there is an integer parameter defined for the page size with a default value of 350. With this value set to 350, the sample search service will retrieve 350 results per page. In the out-of-the-box IBM Content Navigator browse pane or search pane, the user can scroll down the ContentList and when the ContentList has displayed the entire page size of 350 results it will trigger a new search requests to request the next two pages of results. In this section we will add this paging feature in to the custom search plug-in.

The screenshot below shows an example of what is shown to the user when the ContentList widget triggers this paging logic.



With IBM FileNet Content Manager repositories, the FileNet Content Manager API supports the ability to retrieve additional pages of results through the PageIterator object. The logic that invokes this PageIterator class is contained in the original SamplePluginSearchServiceP8 class in the sample plug-in. Below you'll find the relevant code snippet that should go below the PropertyFilter filter line and above the HashMap line.

```
List<Object> searchResults = new ArrayList<Object>(pageSize);
IndependentObjectSet resultsObjectSet = searchScope.fetchObjects(searchSQL,
pageSize, filter, true);
PageIterator pageIterator = resultsObjectSet.pageIterator();
int itemCount = 0;
if (pageIterator.nextPage()) {
        for (Object obj : pageIterator.getCurrentPage()) {
                searchResults.add(obj);
                itemCount++;
        }
}
```

To retrieve the next page of results, we use the pageIterator.nextPage() method. To implement the paging logic, we need to retrieve the pageIterator object from the session and then use it to get the next page when the user triggers the continued search.

The search requests are really divided in to two types:

- The initial search request to get the very first page of results, saving the *pageIterator* object into the session. Store the *continuationData* value into the ResultSet object.

- Retrieve the *pageIterator* object from the session. If a pageIterator object exists, get the next page of results using the pageIterator object.

The initial search request can be implemented by modifying the SamplePluginSearchServiceP8 class. We add the pageIterator object into session. When the first page size is equal to the pageSize value, that tells us there may be more results available. The sessionKey *"*pageIterator*"* is stored in to the result set with a name of *continuationData* and then stored in to the session. The itemCount parameter is the number of results retrieved in the first page. The request parameter is the request object of web application.

Here is the code snippet that should be added to the bottom of the executeP8Search method.

```
String sessionKey = "pageIterator";
request.getSession().removeAttribute(sessionKey);
if (itemCount == pageSize) {
        jsonResultSet.put("continuationData", sessionKey);
        request.getSession().setAttribute(sessionKey, pageIterator);
}   }
```

The ContentList widget already implements the necessary logic to trigger the paging properly. The ContentList widget checks the ResultSet object for the presence of the *continuationData* parameter. When the *continuationData* parameter is present, the ContentList widget will call the /p8/continueQuery service to return the next page of results if one is available. This service is the search service provided by the out-of-the-box Content Navigator mid-tier – this is the service we are replacing with our own custom. In order to trigger the ContentList widget to call your new custom service, you need to leverage the request filter feature supported by IBM Content Navigator. A request filter is a mechanism that allows you to replace any existing Content Navigator service call with our own. In this case, we want to capture the /p8/continueQuery service call made by the ContentList, and use another method to provide our own custom response. We will build a request filter named ContinueQueryService.java to replace the /p8/continueQuery service in our custom plug-in.

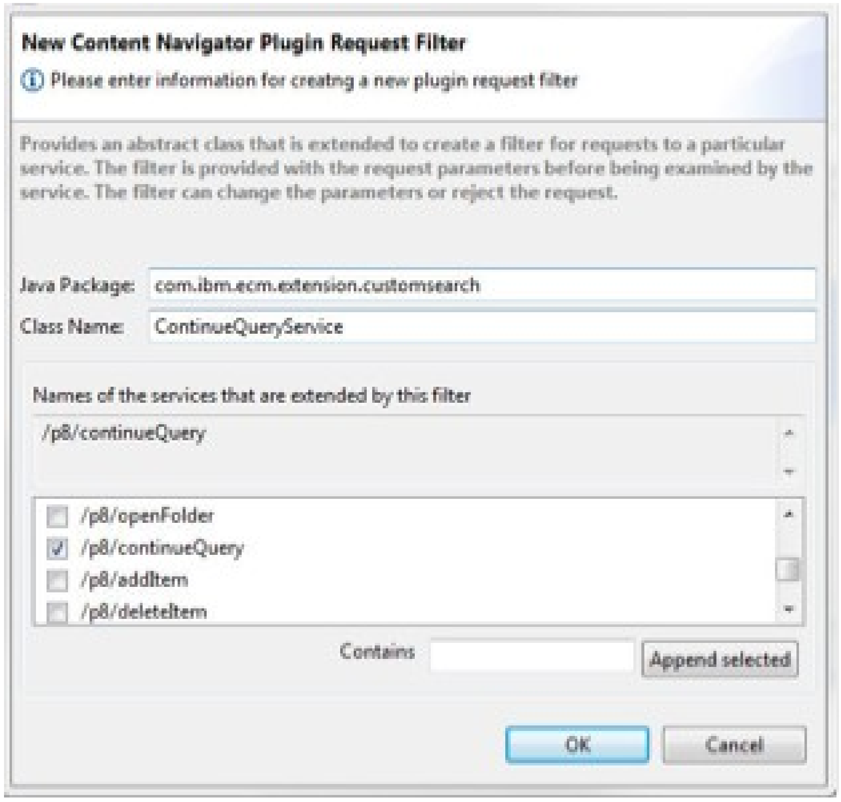
1. Create a new request filter in your Eclipse plug-in. Right click on the **com.ibm.ecm.extension.customsearch** package in the custom search plug-in project, and select **IBM Content Navigator ✹ Server Extensions ✹ New Request Filter**.

2. In the new request filter dialog, configure it as follows

   Enter a Class Name of ContinueQueryService.

   Select the **/p8/continueQuery service** from the services list and click **Append selected**. You can select more than one service to be filtered, but in this case we only want to override this one service.

3. Click **OK**. The request filter class is then automatically generated and properly registered in the main CustomSearchPlugin class.


New Content Navigator Plugin Request Filter

Please enter information for creatng a new plugin request filter

Provides an abstract class that is extended to create a filter for requests to a particular service. The filter is provided with the request parameters before being examined by the service. The filter can change the parameters or reject the request.

Java Package: com.ibm.ecm.extension.customsearch

Class Name: ContinueQueryService

Names of the services that are extended by this filter

/p8/continueQuery

- [ ] /p8/openFolder
- [x] /p8/continueQuery
- [ ] /p8/addItem
- [ ] /p8/deleteItem

In the ContinueQueryService class that you just created, the getFilteredServices() method shows which service(s) are filtered, which in our case is an array with only one element.

public String[] getFilteredServices() {
return new String[] { "/p8/continueQuery" };
}

The primary logic for our request filter is in the filter() method. You can implement any logic you need to within this method but the method must return a JSONObject.

In our example, we will try to retrieve the *continuationData* parameter from the request object and use it as the key to retrieve the pageIterator object from the session. We retrieve this pageIterator object and then use it to get the next page of results for the query. If the pageIterator value is null, that means that the request is not for our custom search plug-in but instead meant to be sent on to the existing IBM Content Navigator service. In this situation, simple return null and the request will be routed to the original IBM Content Navigator service handler.

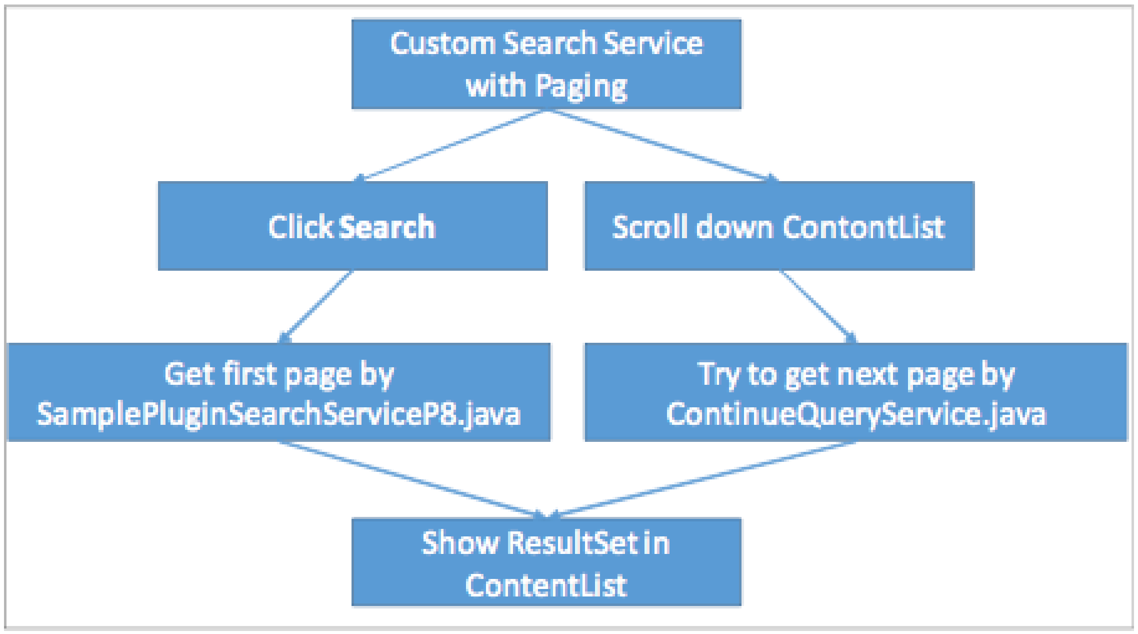
```
        public  JSONObject filter(PluginServiceCallbacks callbacks,
HttpServletRequest request, JSONArtifact jsonRequest) throws Exception {
                String continueQuerySessionKey =
request.getParameter("continuationData");
                if (request.getSession().getAttribute(continueQuerySessionKey) !
= null) {
                    // the request is from the plug-in service
                        JSONResultSetResponse jsonResults = new
JSONResultSetResponse();
                        this.execute(callbacks, request, jsonResults);
                    return jsonResults;
                } else {
                    // this request is not related with the sample plug-in search
service
                    // so return to the default ICN service handler.
                    return null;
               }
          }
```

Note that in our example we will modify the pageSize to 10 to make it simpler to test the paging logic. If you choose to keep the page size set to 350, then you can still test the paging logic but you will need to scroll through 350 results before the ContentList widget will send a request to retrieve an additional page of results.

Copy the following code in to the top of the ContinueQuery class file. After doing so, you will need to right click, select Source, and then Organize Imports to update the import statements to avoid compilation issues.

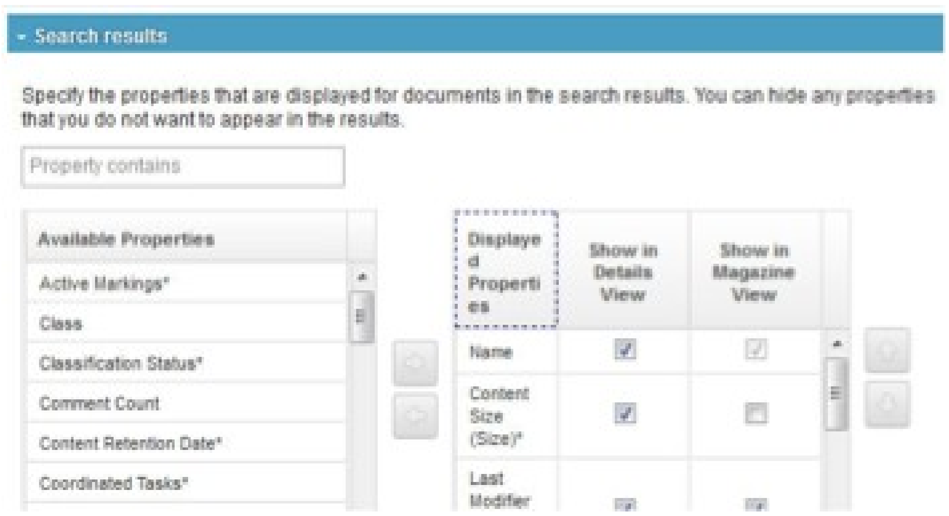
```
public static final String REPOSITORY_ID = "repositoryId";
public static final String REPOSITORY_TYPE = "repositoryType";
public static final int pageSize = 10;

public void execute(PluginServiceCallbacks callbacks,
          HttpServletRequest request, JSONResultSetResponse jsonResults)
          throws Exception {
          String methodName = "execute";
          callbacks.getLogger().logEntry(this, methodName, request);

      String repositoryId = request.getParameter(REPOSITORY_ID);
          String continueQuerySessionKey =
request.getParameter("continuationData");

      jsonResults.setPageSize(pageSize);
          List<Object> searchResults = new ArrayList<Object>(pageSize);
          int itemCount = 0;
          try {
                    Subject subject = callbacks.getP8Subject(repositoryId);
                    UserContext.get().pushSubject(subject);
                    Object synchObject = callbacks.getSynchObject(repositoryId,
"p8");
                    if (synchObject != null) {
                              synchronized (synchObject) {
                                        PageIterator pageIterator = (PageIterator)
request
                                                  .getSession().getAttribute(continue
QuerySessionKey);
                                        if (pageIterator.nextPage()) {
                                                  for (Object obj :
pageIterator.getCurrentPage()) {
                                                            searchResults.add(obj);
                                                            itemCount++;
                                                  }
                                        }
                                        if (itemCount == pageSize) {
                                                  String sessionKey = "pageIterator";
                                                  jsonResults.put("continuationData",
sessionKey);
                                        } else {

          request.getSession().removeAttribute(
          continueQuerySessionKey);
                              }
                        }
                  } else {
                        PageIterator pageIterator = (PageIterator)
request.getSession()
                                    .getAttribute(continueQuerySessionK
ey);
                        if (pageIterator.nextPage()) {
                              for (Object obj :
pageIterator.getCurrentPage()) {
                                    searchResults.add(obj);
                                    itemCount++;
                              }
                        }
                        if (itemCount == pageSize) {
                              String sessionKey = "pageIterator";
                              jsonResults.put("continuationData",
sessionKey);
                        } else {

          request.getSession().removeAttribute(continueQuerySessionKey);
                        }
                  }
                  // Retrieve the privilege masks for the search results.
                  HashMap<Object, Long> privMasks = callbacks.getP8PrivilegeMasks(
                              repositoryId, searchResults);
                  ObjectStore objectStore =
callbacks.getP8ObjectStore(repositoryId);
                  for (Object searchResult : searchResults) {
                        Document doc = (Document) searchResult;
                        /*
                         *  IDs use the form:
                         *  <object class name>,<object store ID>,<object ID>
                         */
                        StringBuffer sbId = new StringBuffer();


sbId.append(doc.getClassName()).append(",").append(objectStore.get_Id().toString())
.append(",").append(doc.get_Id().toString());

                        long privileges = (privMasks != null) ?
privMasks.get(doc) : 0L;

                        JSONResultSetRow row = new
JSONResultSetRow(sbId.toString(), doc.get_Name(), doc.get_MimeType(), privileges);

                        VersionSeries versionSeries = doc.get_VersionSeries();
                        if (versionSeries != null) {
                              Id vsIdId = versionSeries.get_Id();
                              if (vsIdId != null) {
                                    row.put("vsId", vsIdId.toString());
                              }
                        }

                        row.put("template", doc.getClassName());
                        row.put("objectStoreId",
objectStore.get_Id().toString());

                              // Add locked user information (if any)
                              row.addAttribute("locked", doc.isLocked(),
JSONResultSetRow.TYPE_BOOLEAN, null, (new Boolean(doc.isLocked())).toString());
                              row.addAttribute("lockedUser", doc.get_LockOwner(),
JSONResultSetRow.TYPE_STRING, null, doc.get_LockOwner());
                              row.addAttribute("currentVersion",
doc.get_IsCurrentVersion(), JSONResultSetRow.TYPE_BOOLEAN, null, (new
Boolean(doc.get_IsCurrentVersion())).toString());

                              // Add the attributes
                              row.addAttribute("ID", doc.get_Id().toString(),
JSONResultSetRow.TYPE_STRING, null, doc.get_Id().toString());
                              row.addAttribute("className", doc.getClassName(),
JSONResultSetRow.TYPE_STRING, null, doc.getClassName());
                              row.addAttribute("ModifiedBy", doc.get_LastModifier(),
JSONResultSetRow.TYPE_STRING, null, doc.get_LastModifier());
                              row.addAttribute("LastModified",
doc.get_DateLastModified().toString(), JSONResultSetRow.TYPE_TIMESTAMP, null,
doc.get_DateLastModified().toString());
                              row.addAttribute("Version",
doc.get_MajorVersionNumber() + "." + doc.get_MinorVersionNumber(),
JSONResultSetRow.TYPE_STRING, null, doc.get_MajorVersionNumber() + "." +
doc.get_MinorVersionNumber());
                              row.addAttribute("{NAME}",
doc.get_Name(),JSONResultSetRow.TYPE_STRING, null, doc.get_Name());
                              row.addAttribute("ContentSize", doc.get_ContentSize(),
JSONResultSetRow.TYPE_INTEGER, null, null);

                              if
(doc.getProperties().isPropertyPresent(PropertyNames.CM_THUMBNAILS)) {
                                        CmThumbnailSet thumbnails =
doc.get_CmThumbnails();
                                        if (thumbnails != null && !
thumbnails.isEmpty()) {
                                                  Iterator iter =
thumbnails.iterator();
                                                  while (iter.hasNext()) {
                                                            CmThumbnail thumbnailObj =
(CmThumbnail) iter.next();
                                                            byte[] thumbnail =
thumbnailObj.get_Image();

                                                            if (thumbnail != null &&
thumbnail.length > 0) {
                                                                      JSONObject
thumbnailJson = new JSONObject();

          thumbnailJson.put("mimeType", thumbnailObj.get_MimeType());

          thumbnailJson.put("image", "data:" + thumbnailObj.get_MimeType() +
";base64," + Base64.encodeBase64String(thumbnail));

          row.addAttribute("thumbnail", thumbnailJson,
JSONResultSetRow.TYPE_OBJECT, null, "");
                                                            }
                                                  }
                                        }
                                        thumbnails = null;
                                }

                                jsonResultSet.addRow(row);
                        }
        } catch (Exception e) {
                        // provide error information
callbacks.getLogger().logError(this,
                        // methodName, request, e);
                        JSONMessage jsonMessage = new JSONMessage(0, e.getMessage(),
                                        "This error may occur if the search string is
invalid.",
                                        "Ensure the search string is the correct
syntax.",
                                        "Check the IBM Content Navigator logs for
more details.",
                                        "");
                        jsonResults.addErrorMessage(jsonMessage);
        } finally {
                        UserContext.get().popSubject();
                        callbacks.getLogger().logExit(this, methodName, request);
        }
}
```

The logic of custom search service with paging is summarized in Figure 5-17 on page 200.


Custom Search Service with Paging
Click Search
Scroll down ContentList
Get first page by SamplePluginSearchServiceP8.java
Try to get next page by ContinueQueryService.java
Show ResultSet in ContentList

Step 2 – Retrieve the result column settings from the administration configuration

Now that you have created a custom search service that will allow the user to properly page through multiple pages of results, it's time to deal with the columns that the user will see in the ContentList. In the sample plug-in the columns that will be shown to the user in the ContentList are hard-coded. IBM Content Navigator allows administrators to configure the default columns shown to the user in the administration feature as part of the configuration of the Repository definition on the Search tab. In this example, we will update the sample plug-in to retrieve these settings and remove the hard-coded logic that currently exists.


Search results

Specify the properties that are displayed for documents in the search results. You can hide any properties that you do not want to appear in the results.

Available Properties
Active Markings*
Class
Classification Status*
Comment Count
Content Retention Date*
Coordinated Tasks*

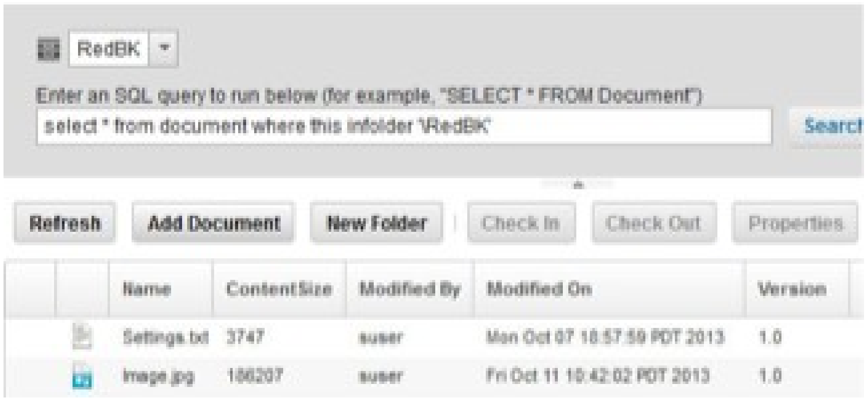
Displayed Properties	Show in Details View	Show in Magazine View
Name	✓	✓
Content Size (Size)*	✓	☐
Last Modifier	✓	✓

Luckily, Content Navigator makes it quite easy to retrieve these settings as you can see from the following code snippet the needs to be at the top of the buildP8ResultStructure method of the SamplePluginSearchServiceP8 class.

```
private static void buildP8ResultStructure(HttpServletRequest request,
JSONResultSetResponse jsonResultSet, MessageResources resources, Locale
clientLocale) {
          RepositoryConfig repoConf = Config.getRepositoryConfig(request);
          String[] folderColumns = repoConf.getSearchDefaultColumns();
```

Once we have retrieved the default search columns, we can use those to decide which properties should be included in the query string thereby improving the efficiency of the query being executed. Here is the complete code for the method.

```
private static void buildP8ResultStructure(HttpServletRequest request,
JSONResultSetResponse jsonResultSet, MessageResources resources, Locale
clientLocale) {
          RepositoryConfig repoConf = Config.getRepositoryConfig(request);
          String[] folderColumns = repoConf.getSearchDefaultColumns();

          String[] states = new String[1];
          states[0] = JSONResultSetColumn.STATE_LOCKED;

          jsonResultSet.addColumn(new JSONResultSetColumn("&nbsp;",
"multiStateIcon", false, states));
          jsonResultSet.addColumn(new JSONResultSetColumn("&nbsp;", "17px",
"mimeTypeIcon", null, false));
          jsonResultSet.addColumn(new
JSONResultSetColumn(resources.getMessage(clientLocale, "search.results.header.id"),
"200px", "ID", null, false));
          jsonResultSet.addColumn(new JSONResultSetColumn("Class Name", "125px",
"className", null, false));
          for (String columnString : folderColumns) {
                    if (columnString.equals("LastModifier"))
                              jsonResultSet.addColumn(new
JSONResultSetColumn(resources.getMessage(clientLocale,
"search.results.header.lastModifiedByUser"), "125px", "ModifiedBy", null, false));
                    else if (columnString.equals("DateLastModified"))
                              jsonResultSet.addColumn(new
JSONResultSetColumn(resources.getMessage(clientLocale,
"search.results.header.lastModifiedTimestamp"), "175px", "LastModified", null,
false));
                    else if (columnString.equals("MajorVersionNumber"))
                              jsonResultSet.addColumn(new
JSONResultSetColumn(resources.getMessage(clientLocale,
"search.results.header.version"), "50px", "Version", null, false));
                    else if (columnString.equals("{NAME}"))
                              jsonResultSet.addColumn(new
JSONResultSetColumn("Name", "200px", columnString, null, false));
                    else {
                              jsonResultSet.addColumn(new
JSONResultSetColumn(columnString, "80px", columnString, null, true));
                    }
          }

          // Magazine view
          jsonResultSet.addMagazineColumn(new JSONResultSetColumn("thumbnail",
"60px", "thumbnail", null, null));

          JSONArray fieldsToDisplay = new JSONArray();
          JSONObject jsonObj = new JSONObject();
          jsonObj.put("field", "className");
          jsonObj.put("displayName", "Class");
          fieldsToDisplay.add(jsonObj);

          jsonObj = new JSONObject();
          jsonObj.put("field", PropertyNames.LAST_MODIFIER);
          jsonObj.put("displayName", resources.getMessage(clientLocale,
"search.results.header.lastModifiedByUser"));
          fieldsToDisplay.add(jsonObj);

          jsonObj = new JSONObject();
          jsonObj.put("field", PropertyNames.DATE_LAST_MODIFIED);
          jsonObj.put("displayName", resources.getMessage(clientLocale,
"search.results.header.lastModifiedTimestamp"));
          fieldsToDisplay.add(jsonObj);

          jsonObj = new JSONObject();
          jsonObj.put("field", PropertyNames.MAJOR_VERSION_NUMBER);
          jsonObj.put("displayName", resources.getMessage(clientLocale,
"search.results.header.version"));
          fieldsToDisplay.add(jsonObj);

          JSONArray extraFieldsToDisplay = new JSONArray();
          jsonObj = new JSONObject();
          jsonObj.put("field", "CmThumbnails");
          jsonObj.put("displayName", "Thumbnails");
          jsonObj.put("decorator",
"MagazineViewDecorator.contentCellDecoratorCmThumbnails");
          extraFieldsToDisplay.add(jsonObj);

          jsonResultSet.addMagazineColumn(new JSONResultSetColumn("content",
"100%", "content", fieldsToDisplay, null));
}
```



Enter an SQL query to run below (for example, "SELECT * FROM Document")

select * from document where this infolder '\RedBK'

Name	ContentSize	Modified By	Modified On	Version
Settings.txt	3747	suser	Mon Oct 07 18:57:59 PDT 2013	1.0
Image.jpg	186207	suser	Fri Oct 11 10:42:02 PDT 2013	1.0

At this point you should build and test the plugin again to make sure it's working properly.

Step 3 – Overview of the primary files in the custom search plug-in

The previous sections covered the most important points of the sample code needed to implement paging and improve query efficiency. There is additional logic contained in the Java classes that was not covered. To help you navigate the code and review it in detail, below you will find a listing all of the Java classes not discussed in detail in the previous sections. These files can be found in the sample project included with the lab.

- CustomSearchPlugin: The main class of the custom search plug-in. This class is required in every plug-in and it generally is not one that you need to touch if you are using the Eclipse plug-in for Content Navigator plugin development. This class simply tells Content Navigator what extensions your plugin will be providing – specifically what features, actions, layouts, request filters, etc.

- SearchService.: This class is the service extension that implements the custom search. The detailed implementation is contained in the SamplePluginSearchServiceP8 class.

- SamplePluginSearchServiceP8: This class is copied from the sample plug-in and provides the FileNet P8 specific search logic, including the advanced paging support.

- ContinueQueryService: a request filter that implements our custom paging logic when the
C ContentList calls the /p8/continueQuery service

•

7. Lab Activity 8 – Creating a custom tree feature

This chapter describes how to create a new feature plug-in that will include the custom search services from chapter 7 as well as adding some custom widgets. Specifically, in this activity you'll add a virtual folder browse pane leveraging the custom search services you've already built.

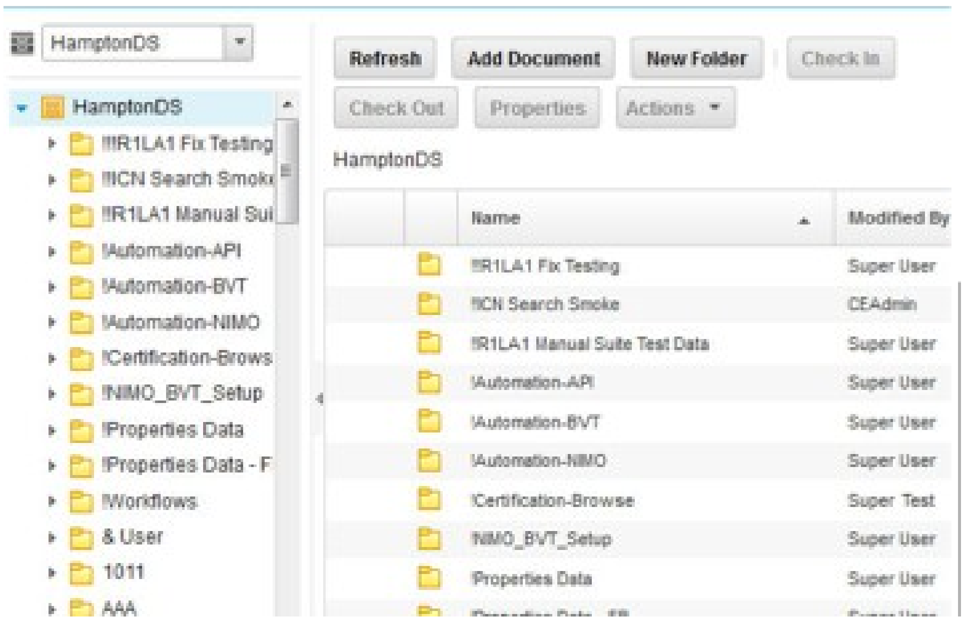
This chapter covers the following topics:

- Overview of the Exercise
- Creating a custom layout for the feature
- Creating a tree widget to display a custom tree
- Add logic to the tree to execute the search and display the results
- Adding modules to the ContentList widget

8. Overview

In the previous activity, we created a new feature with custom search services that was imported from the existing Content Navigator sample plug-in. However, we have not yet actually leveraged this custom search service in our own feature page. The CustomSearchPane was a simple page created to demonstrate that the search service was working properly. In this chapter, we will now enhance the custom search plug-in to address a more real-world scenario by building a *virtual folder browse pane*.

IBM Content Navigator provides a feature called the browse pane. This feature allows users to browse the repository folder tree. Clicking on a node in the folder tree will cause the contents of that folder to be shown in the ContentList in the right-hand pane.



Name	Modified By
!!R1LA1 Fix Testing	Super User
!ICN Search Smoke	CEAdmin
!R1LA1 Manual Suite Test Data	Super User
!Automation-API	Super User
!Automation-BVT	Super User
!Automation-NIMO	Super User
!Certification-Browse	Super Test
!NIMO_BVT_Setup	Super User
!Properties Data	Super User

In this example, we will create an alternative browse feature called **virtual folders**. There will still be a tree, but the tree will not be based on folders: instead some nodes will be document classes and some nodes will be document property values.

Users will be able to build a navigation tree themselves. When a user clicks on a node in the tree, there will be a search triggered to retrieve the matching documents. Basically, we will leave the definition of the tree up to the user. The user can define a personalized tree structure and then save that tree structure configuration.

In this chapter, the tree structure is hard coded, but you can certainly choose to enhance it to add more features later. We will enhance the custom search plug-in from the previous activity to support this new virtual folder tree feature.

9. Creating a custom layout

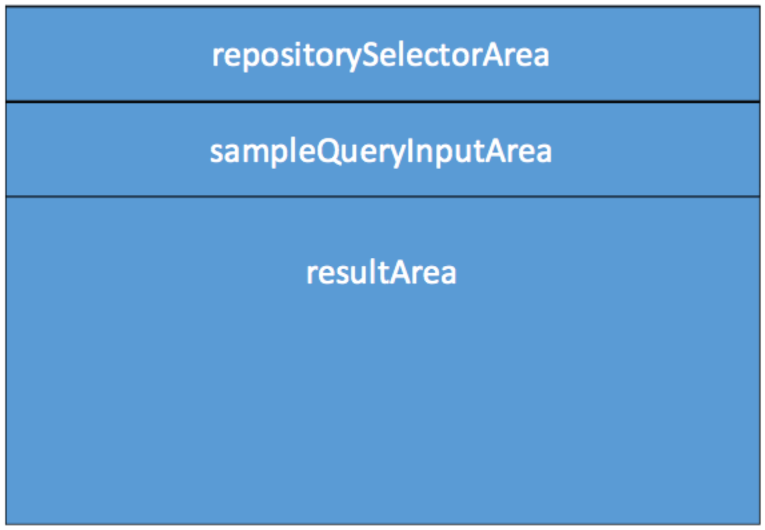
For every Dojo widget, there is a corresponding HTML template defined. The layout of the widget is defined in this template file. The dijit._TemplatedMixin helps you to create widgets quickly and easily. IBM Content Navigator uses this Dojo dijit to build widgets.

When you created the custom feature pane for the custom search plug-in in the previous activity, there was a template HTML file created automatically by the Content Navigator Eclipse plug-in. The name of the file is CustomFeaturePane.html and the code for the feature is shown below.

```
<div class="ecmCenterPane"><div data-dojo-type="idx/layout/BorderContainer"

data-dojo-props="gutters:true"><div data-dojo-attach-point="searchArea"

data-dojo-type="dijit/layout/ContentPane" data-dojo-props="splitter:true,region:'top'"

class="sampleSearchArea"><div data-dojo-attach-point="repositorySelectorArea"

class="sampleRepositorySelectorArea"></div> <label for="${id}_queryString"

data-dojo-attach-point="cm8HelpText" style="display:none">Enter an XPath query to run below (for
example, &quot;/NOINDEX&quot;)</label>

<label for="${id}_queryString" data-dojo-attach-point="p8HelpText" style="display:none">Enter an SQL
query to run below (for example, &quot;SELECT * FROM Document&quot;)</label>

                <div class="sampleQueryInputArea">
                  <input id="${id}_queryString"
data-dojo-attach-point="queryString" data-dojo-type="dijit/form/TextBox"></input>

<button data-dojo-attach-point="searchButton" data-dojo-type="dijit/form/Button"

data-dojo-attach-event="onClick: runSearch" class="searchButton">

         ${messages.search}
       </button>
     </div>
```

```
    </div>
<div data-dojo-attach-point="resultsArea" data-dojo-type="dijit/layout/ContentPane" data-dojo-
props="region:'center'">

<div data-dojo-attach-point="searchResults" data-dojo-type="ecm/widget/listView/ContentList"

data-dojo-props="emptyMessage:'${messages.folder_is_empty}'"> </div>

    </div>
  </div>
</div>
```

The layout of this template is shown below.



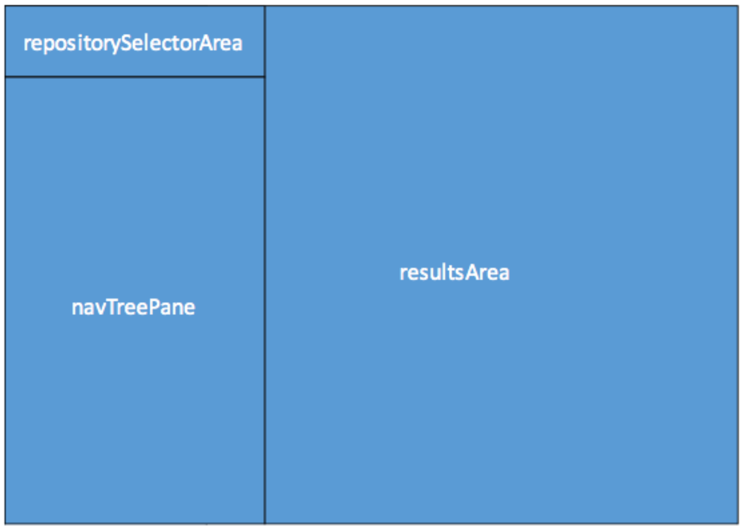
The following is a description of the components in the template:

- **data-dojo-attach-point** is the reference to the DOM element created. This value can be used in the widget as a property. For example, you can find **cm8HelpText** in CustomFeaturePane.js by using **this.cm8HelpText**.

- **data-dojo-type** provides a reference to a pre-defined Dojo widget. For example, the search button uses the existing dijit/form/Button widget.

- **data-dojo-attach-event** provides the mechanism to handle an event in the widget. For example, the **onClick** event of the search button will be handled by the **runSearch** function in CustomFeaturePane.js.

- **data-dojo-props** defines properties of the widget.The layout we will use for the new feature pane o of the virtual folder browsing feature will be similar to the IBM Content Navigator browse pane. The figure below shows an example of this layout.



In this activity, you'll modify the template HTML file to match the layout in the box above.

To create this layout, we need to create a new feature in the custom search plug-in.

1. Right click on the **com.ibm.ecm.extension.customsearch** package in the CustomSearchPlugin project in eclipse and select **IBM Content Navigator ✹ New Feature....**

2. For the class name, enter **VirtualFolderBrowePane**.

3. Any 32*32 PNG file will work for the launch icon class. In this case, let's select a new file named **search.png** from the local directory C:\Navigator\samplePlugins\Lab 6_6\CustomSearchPlugin\src\com\ibm\ecm\extension\customsearch\WebContent\images. Make sure the name of the class is CustomSearchPluginLaunchIcon and then Click **OK**.

4. After you click OK, the wizard will create the following files:

   a. VirtualFolderBrowsePane.java

   b. VirtualFolderBrowsePane.js

   c. VirtualFolderBrowsePane.html files

5. Check the getFeatures() method in the CustomSearchPlugin.java file to ensure that it looks like the following code snippet.

```
           public com.ibm.ecm.extension.PluginFeature[] getFeatures() {
              return new com.ibm.ecm.extension.PluginFeature[] {
                                   new
com.ibm.ecm.extension.customsearch.CustomFeaturePane(),
                                   new
com.ibm.ecm.extension.customsearch.VirtualFolderBrowsePane()};
            }
```

6. TheVirtualFolderBrowsePane.html generated by the Content Navigator Eclipse plug-in contains no

content except for an empty div element. Replace the content of this file with the code below.

```
<div class="ecmCenterPane" data-dojo-attach-point="containerNode">
        <div data-dojo-type="idx.layout.BorderContainer" data-dojo-attach-
point="container" class="contentPane" gutters="false"  design="sidebar">
                <div data-dojo-type="dijit.layout.ContentPane"
region="leading" class="paneNoOverflow" splitter="true">
                        <div data-dojo-
type="dijit.layout.BorderContainer" gutters="false" class="navContainer">
                                <div data-dojo-
type="dijit.layout.ContentPane" data-dojo-props="region:'top'" >
                                <div data-dojo-attach-
point="repositorySelectorArea" class="sampleRepositorySelectorArea"></div>

                            </div>
                                <div data-dojo-attach-
point="searchSelectorArea" class="navContainerBottomBar" data-dojo-
type="dijit.layout.ContentPane" region="center">
                                        <div data-dojo-
type="dijit.layout.ContentPane" id="navTreePane" ,
style="width:100%;height:100%;overflow:auto;", data-dojo-attach-
point="navTreePaneObj" ></div>
                                </div>
                        </div>
                </div>
                <div data-dojo-type="dijit.layout.ContentPane"
region="center">
                        <div data-dojo-attach-point="navResult" data-
dojo-type="ecm.widget.listView.ContentList" emptyMessage="folder is
empty"></div>
                </div>
        </div>
</div>
```

7. The feature icon file will be copied to the com.ibm.ecm.extension.customsearch.WebContent.images package. You can also manually add icons to that package if you choose. Then the icon definition will be in the CustomSearchPlugin.css file.

10. Creating a custom tree widget

In the previous section we created a custom layout for our new feature. The repository selector code is already contained in the CustomFeaturePane.js file and we can re-use it in VirtualFolderBrowsePane.js. Overwrite the code in VirtualFolderBrowsePane.js with the following code. After doing so, we'll walk through some of the important details.

```
define([
        "dojo/_base/declare",
        "dojo/_base/lang",
        "dojo/dom-construct",
                "idx/layout/BorderContainer",
                "dijit/layout/ContentPane",
                "dojo/json",
                "dojo/store/Memory",
   "dijit/tree/ObjectStoreModel",
                "dijit/Tree",
                "ecm/model/Request",
                "ecm/model/ResultSet",
                "ecm/widget/layout/_LaunchBarPane",
                "ecm/widget/layout/_RepositorySelectorMixin",
                "ecm/widget/listView/ContentList",
                "ecm/widget/listView/gridModules/RowContextMenu",
                "ecm/widget/listView/modules/Toolbar",
                "ecm/widget/listView/modules/Breadcrumb",
                "ecm/widget/listView/modules/InlineMessage",
                "ecm/widget/listView/modules/TotalCount",
                "ecm/widget/listView/modules/FilterData",
                "ecm/widget/listView/modules/DocInfo",
                "ecm/widget/listView/gridModules/DndRowMoveCopy",
                "ecm/widget/listView/gridModules/DndFromDesktopAddDoc",
                "ecm/widget/listView/modules/Bar",
                "ecm/widget/listView/modules/ViewDetail",
                "ecm/widget/listView/modules/ViewMagazine",
                "ecm/widget/listView/modules/ViewFilmStrip",
                "dojo/text!./templates/VirtualFolderBrowsePane.html"
],

function(declare,
                                lang,
                                domConstruct,
                                idxBorderContainer,
                                ContentPane,
                                json,
                                Memory,
                                ObjectStoreModel,
                                Tree,
                                Request,
                                ResultSet,
                                _LaunchBarPane,
                                _RepositorySelectorMixin,
                                ContentList,
                                RowContextMenu,
                                Toolbar,
                                Breadcrumb,
                                InlineMessage,
                                TotalCount,
                                FilterData,
                                DocInfo,
                                DndRowMoveCopy,
                                DndFromDesktopAddDoc,
                                Bar,
                                ViewDetail,
                                ViewMagazine,
                                ViewFilmStrip,
                                template) {

                /**
                 * @name customSearchPluginDojo.VirtualFolderBrowsePane
                 * @class
                 * @augments ecm.widget.layout._LaunchBarPane
                 */
                return declare("customSearchPluginDojo.VirtualFolderBrowsePane", [
                                _LaunchBarPane,
                                _RepositorySelectorMixin
                ], {
                                /** @lends customSearchPluginDojo.VirtualFolderBrowsePane.prototype */

                                templateString: template,
                                widgetsInTemplate: true,
                                classnames:[],
                                mainfolders:[],
                                createdTreeItems:false,

                                postCreate: function() {
                                            this.logEntry("postCreate");
                                            this.inherited(arguments);
                                            this.defaultLayoutRepositoryComponent = "others";

                                            this.setRepositoryTypes("cm,p8");
                                            this.createRepositorySelector();
                                            this.doRepositorySelectorConnections();

                                            // If there is more than one repository in the list, show the selector to
the user.
                                            if (this.repositorySelector.getNumRepositories() > 1) {
                                                        domConstruct.place(this.repositorySelector.domNode,
this.repositorySelectorArea, "only");
                                            }
                                            this.logExit("postCreate");
                                },

                                /**
                                 * Returns the content list grid modules used by this view.
                                 *
                                 * @return Array of grid modules.
                                 */
                                getContentListGridModules: function() {
                                            var array = [];
                                            array.push(DndRowMoveCopy);
                                            array.push(DndFromDesktopAddDoc);
                                            array.push(RowContextMenu);
                                            return array;
                                },
		/**
		 * Returns the content list modules used by this view.
		 *
		 * @return Array of content list modules.
		 */
		getContentListModules: function() {
			var viewModules = [];
			viewModules.push(ViewDetail);
			viewModules.push(ViewMagazine);
			if (ecm.model.desktop.showViewFilmstrip) {
				viewModules.push(ViewFilmStrip);
			}

			var array = [];
			array.push(DocInfo);
			array.push({
				moduleClass: Bar,
				top: [
						[
							[
								{
moduleClass: Toolbar
								},
								{
moduleClass: FilterData
								},
								{
moduleClasses: viewModules,

"className": "BarViewModules"
								}
							]
						],
						[
							[
								{
moduleClass: Breadcrumb,

rootPrefix:  "Virtual folder"
								}
							]
						],
						[
							[
								{
moduleClass: InlineMessage,
                "className": "inlineMessage"
                                                                                    }
                                                                            ]
                                                                    ]
                                                            ],
                                                            bottom: [
                                                                    [
                                                                            [
                                                                                    {

                moduleClass: TotalCount
                                                                                    }
                                                                            ]
                                                                    ]
                                                            ]
                                            });
                                            return array;
                            },

                            /**
                             * Loads the content of the pane. This is a required method to insert a pane into
the LaunchBarContainer.
                             */
                            loadContent: function() {
                                            this.logEntry("loadContent");
                                            if (!this.repository) {
                                                            this.setPaneDefaultLayoutRepository();
                                            }
                                            var data ="{\"name\": \"Multiple Demension
Tree\",\"id\": \"root\",\"children\": [{\"name\": \"My
Navigator\",\"id\": \"my_navigator\", \"children\":[]}]}";
                                            this.TreeStore = new Memory({
                                    data: [ json.parse(data) ],
                                    getChildren: lang.hitch(this,function(object){
               return object.children;
                                    })
                              });
                                            this._resetTree();
                                            this.navTree.placeAt(dijit.byId("navTreePane").containerNode);

                            var callbackGetFolders = lang.hitch(this, function(resultset)
                            {
                            this.mainfolders=[];
                                            for(row in resultset.items)
                                            {
                                                            var element= {
                                                                                    value:resultset.items[row].id,
                                                                                    label:resultset.items[row].name,
                                                                                    name:"Folder:
"+resultset.items[row].name,

                                                                                    id:resultset.items[row].id,
                                                                                    criterionType:"Folder",
                                                            children:[]
                                        }
                                        this.mainfolders.push(element);
                                }
                                this.setupNavTree();
                        });
                        var callbackGetClasses = lang.hitch(this, function(contentClasses)
                        {
                                        this.classnames=[];
                                for(docclass in contentClasses)
                                {
                                        var element= {

                value:contentClasses[docclass].id,

                label:contentClasses[docclass].name,
                                                            name:"Class:
"+contentClasses[docclass].name,
                                                            id:contentClasses[docclass].id,
                                                            criterionType:"Class",
                                                            children:[]

                                        }
                                        this.classnames.push(element);
                                }
                        });

                                if (this.repository && this.repository.canListFolders()) {
                                        var rootItemId = this.repository.rootFolderId || "/";
                                        var _this = this;

                this.repository.retrieveContentClasses(callbackGetClasses);
                                        this.repository.retrieveItem(rootItemId, lang.hitch(this,
function(rootFolder) {
                                                this.repository.rootFolder=rootFolder;
                                                rootFolder.retrieveFolderContents(true,
callbackGetFolders);
                                        }), null, null, null, this._objectStore ? this._objectStore.id :
"");
                                }
                                this.isLoaded = true;
                                this.needReset = false;
                                this.logExit("loadContent");
                        },
                        _resetTree:function()
                        {
                                if(this.navTree)
                                        this.navTree.destroy();
                                TreeModel = new ObjectStoreModel({
                           store: this.TreeStore,
                           query: {id: 'root'},
                           mayHaveChildren: function(item){
```

```
                                        return "children" in item;
                                    }
                                  });
                                            this.navTree = new Tree({
                                                        model: TreeModel,
                                                        onOpenClick: true,
                                                        persist: false,
                                                        getIconClass: lang.hitch(this, function(item,opened)
                                                                    {
                                                                                if(item.id != "root" && item.id !
="my_navigator")
                                                                                            return (opened ?
"searchFolderOpenIcon" : "searchFolderCloseIcon");
                                                                                else
                                                                                            return (opened ?
"dijitFolderOpened" : "dijitFolderClosed");
                                                                    })
                                            }, "divTree");
                                            this.navTree.domNode.style.height="100%";
                                            this.navTree.placeAt(dijit.byId("navTreePane").containerNode);
                                            this.createdTreeItems=false;
                                            this.connect(this.navTree, "onClick", lang.hitch(this, function(item) {
                                                         if(item.id != "root" && item.id !="my_navigator")
                                                        {
                                                                    this.executeSearch(item);
                                                        }
                                            }));
                                },
                                setupNavTree:function()
                                {
                                            if(!this.createdTreeItems)
                                            {
                                                        var firstLayer=this.mainfolders;
                                                        var secondlayer=this.classnames;
                                                        if(secondlayer.length>0)
                                                        {
                                                                    for(j in firstLayer)
                                                                    {

                    firstLayer[j].children=secondlayer;
                                                                    }
                                                        }

                    this.TreeStore.data[0].children[0].children=firstLayer;//it's the path of "my navigator";
                                                        this.createdTreeItems=true;
                                            }
                                },
                                executeSearch:function(item){
                                            var node = this.navTree.getNodesByItem(item);
                                            var path = node[0].tree.path;
                                            var docClassName="";
                                            var mainFolderID="";
                                            for(i=2;path[i]!=undefined;i++)
                                            {
                                                        if(path[i].criterionType=="Class")
                                                        {
                                                                    docClassName=path[i].value;
                                                        }else if(path[i].criterionType == "Folder")
                                                        {
                                                                    mainFolderID=path[i].value;
                                                        }
                                            }
                                            this.runSearch(docClassName,mainFolderID);
                                },
                                /**
                                 * Sets the repository being used for search.
                                 *
                                 * @param repository
                                 *                                      An instance of {@link ecm.model.Repository}
                                 */
                                setRepository: function(repository) {
                                            this.repository = repository;
                                            if (this.repositorySelector && this.repository) {
                                                        this.repositorySelector.getDropdown().set("value",
this.repository.id);
                                            }
                                            this.navResult.reset();
                                            this.loadContent();
                                },
                                runSearch:function(docClassName,mainFolderID,attributeName,attributeValue){
                                            var requestParams = {};
                                            requestParams.repositoryId = this.repository.id;
                                            requestParams.repositoryType = this.repository.type;
                                            if( this.repository.type == "cm" ){
                                                        var scoperule="/* ";
                                                        var baserulestart='[(@SEMANTICTYPE IN (1))';
                                                        var ruleEnd="]"
                                                        var folderrule='INBOUNDLINK[@LINKTYPE =
"DKFolder"]/@SOURCEITEMREF = ';
                                                        var attributerule="";
                                                        if(docClassName!="")
                                                                    scoperule='/'+docClassName+" ";
                                                        var query = scoperule+baserulestart;
                                                        if(attributeName!="" && attributeName!=undefined)
                                                        {
                                                                    attributerule='((@' +attributeName+" =
"+attributeValue +'))'
                                                                    query = query +" AND "+attributerule;
                                                        }
                                                        if(mainFolderID!="")
                                                        {
                             itemid =mainFolderID.split(" ")[6];
                             mainFolderID =itemid.substr(0, itemid.length-2);
                                                                    folderrule = folderrule+'"'+mainFolderID+'"';
```

```
                                                query = query +" AND "+folderrule;
                                        }
                                        query +=ruleEnd;
                                        requestParams.query = query;
                                }else if(this.repository.type=="p8"){
                                        var query = "Select * from ";
                                        if ( docClassName && docClassName.length > 0 ){
                                                query += docClassName;
                                        }else{
                                                query += "DOCUMENT ";
                                        }
                                        if( mainFolderID || ( attributeName && attributeValue) ){
                                                query += " where "
                                        }
                                        if( mainFolderID && mainFolderID.length >0 ){
                                                var folderID =
mainFolderID.substr(  mainFolderID.length-38, mainFolderID.length );
                                                query += " this INFOLDER " + folderID ;
                                        }
                                        requestParams.query=query;
                                }


                                Request.invokePluginService("CustomSearchPlugin", "SearchService",
                                        {
                                                requestParams: requestParams,
                                                requestCompleteCallback: lang.hitch(this,
function(response) {      // success
                                                        response.repository =
this.repository;
                                                        var resultSet = new
ResultSet(response);

        this.navResult.setContentListModules(this.getContentListModules());

        this.navResult.setGridExtensionModules(this.getContentListGridModules());

        this.navResult.setResultSet(resultSet);

                                                        var inlineMessageModule =
this.navResult.getContentListModule( "inlineMessage" );
                                                        if (inlineMessageModule){

        inlineMessageModule.clearMessage();

        inlineMessageModule.setMessage("Result set items length is: " +resultSet.items.length,
"info");
                                                        }
                                                })
                                        }
                                );
                        }
```

```
            });
});
```

One important step is to create the custom tree widget using the Dojo Tree dijit. The Tree dijit needs to have a TreeStore to manage data and nodes that are initialized with a JSON string.

There are three properties for each node to be displayed:

1. name: Display name

2. id: ID for the node

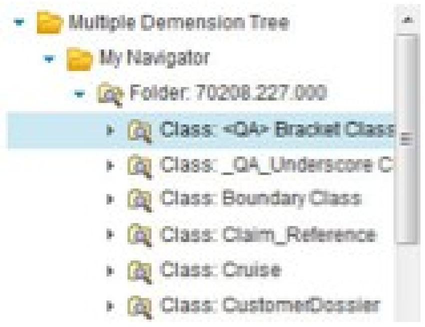
3. children: data that represents the child nodes. See below for an example of the JSON string of a child node.

var data ="{\"name\": \"Multiple Demension Tree\",\"id\": \"root\",\"children\": [{\"name\": \"My Navigator\",\"id\": \"my_navigator\", \"children\":[]}]}";

The tree needs an object store model for servicing the display and managing the data. We will use a Dojo memory store for this purpose. The following code is in the loadContent function.

```
this.TreeStore = new Memory({
                          data: [ json.parse(data) ],
                          getChildren: lang.hitch(this,function(object){
                      return object.children;
                          })
                       });
```

We will create a sample tree to display folders and classes combined together. With this new tree, users can easily navigate documents by folders and / or by classes.



To build this tree, we need to get the folder list and class list from the ECM repository. IBM Content Navigator provides a model layer API that you can use to retrieve this data from the repositories.

To retrieve the list of folders from the root folder requires the root folder id. Using an IBM Content Navigator API as shown below can retrieve the root folder id. The following code snippet is further down in the loadContent function.

var rootItemId = this.repository.rootFolderId || "/";

With the root folder id in hand, you can easily retrieve the top level folder by calling the IBM Content Navigator model API as shown below.

```
this.repository.retrieveContentClasses(callbackGetClasses);

this.repository.retrieveItem(rootItemId, lang.hitch(this, function(rootFolder) {
          this.repository.rootFolder=rootFolder;
          rootFolder.retrieveFolderContents(true, callbackGetFolders);
          }), null, null, null, this._objectStore ? this._objectStore.id : "");
```

You can use the rootFolder.retrieveFolderContents method to retrieve all sub folders of the root folder. With the callbackGetFolders method, you store the folders in to tree nodes.

```
 var callbackGetFolders = lang.hitch(this, function(resultset) {
     this.mainfolders=[];
     for(row in resultset.items) {
          var element= {
                    value:resultset.items[row].id,
                    label:resultset.items[row].name,
                    name:"Folder: "+resultset.items[row].name,
                    id:resultset.items[row].id,
                    criterionType:"Folder",
                    children:[]
          }
          this.mainfolders.push(element);
    }
    this.setupNavTree();
});
```

The code to get the classes is this.repository.retrieveContentClasses(callbackGetClasses);

The callbackGetClasses is a callback function that inserts class data in to the tree. This code is shown below.

```
var callbackGetClasses = lang.hitch(this, function(contentClasses) {
    this.classnames=[];
    for(docclass in contentClasses) {
              var element= {
                    value:contentClasses[docclass].id,
                    label:contentClasses[docclass].name,
                    name:"Class: "+contentClasses[docclass].name,
                    id:contentClasses[docclass].id,
                    criterionType:"Class",
                    children:[]

              }
              this.classnames.push(element);
    }
});
```

To display the classes that the end user wants to display in the custom tree, you can add some filtering logic in to this code.

You may notice that the icon of the custom tree is not the same as the IBM Content Navigator folder icon. In this example, we will show how to use a different icon and how to create the tree with an object store. The code snippet below shows you how to override the getIconClass() method of the tree. This logic will return different icons for different types of nodes. This example also shows you how to use the new icon resource in your tree.

```
this.navTree = new Tree({ model: TreeModel,
                    onOpenClick: true,
                    persist: false,
                    getIconClass: lang.hitch(this, function(item,opened) {
                    if(item.id != "root" && item.id !="my_navigator")
                              return (opened ? "searchFolderOpenIcon" :
"searchFolderCloseIcon");
                    else
                              return (opened ? "dijitFolderOpened" :
"dijitFolderClosed");
                    })
}, "divTree");
```

Note: Not all resource file types are supported by IBM Content Navigator plug-in mechanism. Supported file types are .png, .jpg, .jpeg, .gif, .json, .js, .css, .xml, and .html.

In JavaScript code, the icon string is a class name in the project css file. The icon classes definition in the css file is shown in in the snippet below. The icon files are stored in the images sub-folder of the plug-in, com.ibm.ecm.extension.customsearch.WebContent.images. You can copy these images in to your project from the local file system. The image files can be found in C:\Navigator\samplePlugins\Lab 6_6\CustomSearchPlugin\src\com\ibm\ecm\extension\customsearch\WebContent\images

```
.searchFolderOpenIcon,
        searchFolderOpenIcon {
background-image: url(images/SearchFolderOpened.png); width: 16px;height: 16px;

        }
        .searchFolderCloseIcon,
        searchFolderCloseIcon {
background-image: url(images/SearchFolderClosed.png); width: 16px;height: 16px;

}
```

This example showed you how to build a very simple custom tree. Using this example, you could quite easily build your own tree with other custom data. For example, you might want to build a tree with values from a given document property and then allow users to navigate through different documents by their property values. You could even allow the user to navigator by property values using a hierarchical tree structure.

11. Add logic to the tree to execute the search and display the results

So now that we've created a tree that appears when the repository selector is changed, it's time to define what actions will occur when a node in the tree is selected.

In IBM Content Navigator, when users click on a node in the folder tree, a search request is sent to the repository to retrieve that folder's content that then is displayed in the ContentList in the right hand pane.

For our example, we need similar functionality to the existing IBM Content Navigator browse pane. If the user clicks a node in our custom tree, there will be a search triggered and the results will be displayed in the ContentList. The search criteria will be built according to the properties of the node.

In order to support this behavior, we need to connect the search method to the custom tree. We do this by using the Dojo connect concept which allows us to bind a DOM event to a particular function. In the code snippet below, the onClick event of the custom tree node is bound to the executeSearch() method. The executeSearch() method will build a query string based on the node information and then execute.

```
this.connect(this.navTree, "onClick", lang.hitch(this, function(item) {
          if(item.id != "root" && item.id !="my_navigator"){
                  this.executeSearch(item);
          }
}));
```

In order to execute the search, we will use the existing search service in the custom search plug-in we built in the previous activity. When a user clicks on a folder node, the search will pass the folder id as a parameter to find all documents in that folder. When a user clicks on a class node, the search will pass both the folder id and the class id as the search criteria. After building the parameters, the runSearch() method will be launched to execute the search against the repository.

```
executeSearch:function(item){
          var node = this.navTree.getNodesByItem(item);
          var path = node[0].tree.path;
          var docClassName="";
          var mainFolderID="";
          for(i=2;path[i]!=undefined;i++){
          if(path[i].criterionType=="Class"){
                   docClassName=path[i].value;
          }else if(path[i].criterionType == "Folder"){
                   mainFolderID=path[i].value;
          }
}
this.runSearch(docClassName,mainFolderID);
```

In the previous activity, the user needed to manually enter the entire query string in order to execute a search. However, in this activity the search will be triggered by the user clicking on a tree node, so the query string needs to be generated on the server in our Java code.

```
var query = "Select * from ";
if ( docClassName && docClassName.length > 0 ){
         query += docClassName;
}else{
         query += "DOCUMENT ";
}
if( mainFolderID || ( attributeName && attributeValue) ){
         query += " where "
}
if( mainFolderID && mainFolderID.length >0 ){
         var folderID = mainFolderID.substr(  mainFolderID.length-38,
mainFolderID.length );
         query += " this INFOLDER " + folderID ;
}
requestParams.query=query;
```

With this code now added to our sample, when a user clicks on a node in the custom tree, a search is executed and the search results will be shown in the ContentList.

12. Adding modules to the ContentList widget

As discussed in the previous activity, the ContentList widget is a very important widget in IBM Content

Navigator. The ContentList is based on the Dojo gridx widget with many additional enhancements that are surfaced as pluggable modules.

We described these modules in the previous activity. In this section, we will show you the relevant code where more modules were added to our custom feature's ContentList.

The available modules include: ViewDetail, ViewMagazine, ViewFilmStrip , DocInfo, Toolbar, Bar, DndRowMoveCopy, DndFromDesktopAddDoc , RowContextMenu

The Bar module allows users to arrange ContentList modules below or above the ContentList grid. In the Content Navigator sample plug-in code, there is one line of modules set above the ContentList which contains the ToolBar; the second line contains the view modules for the ContentList. The "*className*"*:* "*BarViewModules*" assigns a class name to the view modules. You can set any class name here. The Bar module will process the string to <table> <tr> & <td> where the array determines how many <table>, <tr>, & <td> items to create.

We can of course add more lines and more modules. In our example, we'll add the FilterData module in the first line. In the second line, we'll add the Breadcrumb module, and in the third line will add the InlineMessage module. Finally, the TotalCount module is added to the bottom of the ContentList grid.

```
          getContentListModules: function() {
          var viewModules = [];
          viewModules.push(ViewDetail);
          viewModules.push(ViewMagazine);
          if (ecm.model.desktop.showViewFilmstrip) {
                    viewModules.push(ViewFilmStrip);
          }

          var array = [];
          array.push(DocInfo);
          array.push({
                    moduleClass: Bar,
                    top: [[[{moduleClass: Toolbar},
                              {moduleClass: FilterData},
                              {moduleClasses: viewModules,"className":
"BarViewModules"}]],
                              [[{moduleClass: Breadcrumb,rootPrefix:  "Virtual
folder"}]],
                              [[{moduleClass: InlineMessage,"className":
"inlineMessage"}]]
                    ],
                    bottom: [[[{moduleClass: TotalCount}]]]
                    });
          return array;},
```

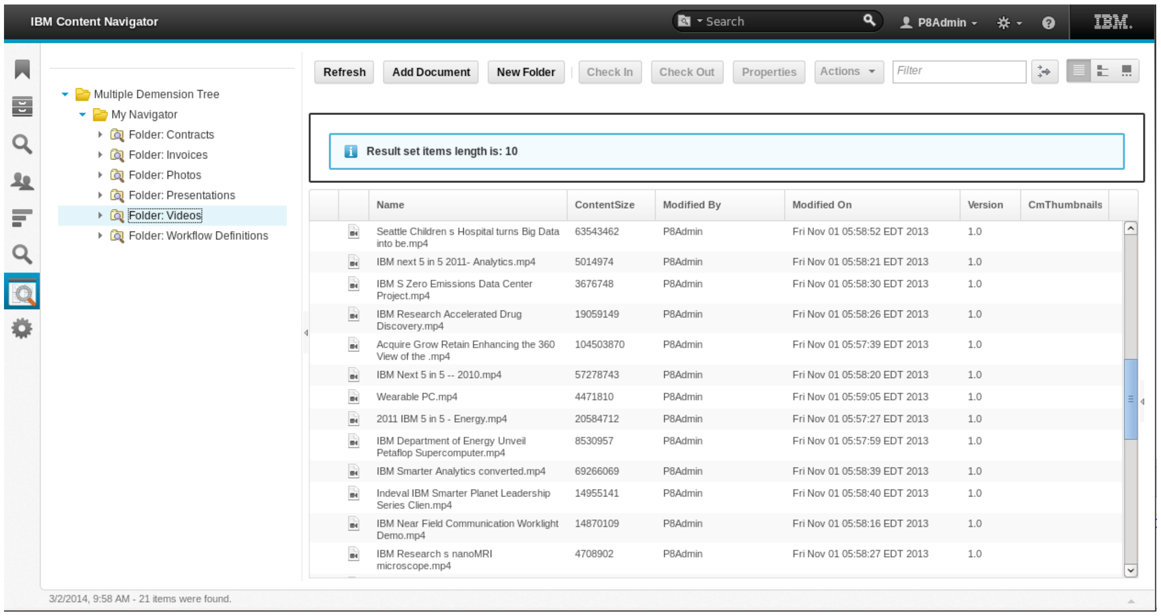
After modifying the code, build and test the plug-in. In order to test the additional feature, you'll need to follow these basic steps.

1. Log in to the ICN administration.

2. Go to the plugins view and reload the plugin. Validate that the CustomFeaturePane now shows up in the list of features that the plugin exposes.

3. Go to the desktops list and edit the Demo desktop. Add the CustomFeaturePane to the desktop displayed features list on the Appearance tab.

4. Log out, refresh the browser, and then log back in to the default desktop at http://base-win2k8x64:9080/navigator.



IBM Content Navigator

Refresh | Add Document | New Folder | Check In | Check Out | Properties | Actions

- Multiple Demension Tree
  - My Navigator
    - Folder: Contracts
    - Folder: Invoices
    - Folder: Photos
    - Folder: Presentations
    - Folder: Videos
    - Folder: Workflow Definitions

Result set items length is: 10

Name	ContentSize	Modified By	Modified On	Version	CmThumbnails
Seattle Children s Hospital turns Big Data into be.mp4	63543462	P8Admin	Fri Nov 01 05:58:52 EDT 2013	1.0	
IBM next 5 in 5 2011- Analytics.mp4	5014974	P8Admin	Fri Nov 01 05:58:21 EDT 2013	1.0	
IBM S Zero Emissions Data Center Project.mp4	3676748	P8Admin	Fri Nov 01 05:58:30 EDT 2013	1.0	
IBM Research Accelerated Drug Discovery.mp4	19059149	P8Admin	Fri Nov 01 05:58:26 EDT 2013	1.0	
Acquire Grow Retain Enhancing the 360 View of the .mp4	104503870	P8Admin	Fri Nov 01 05:57:39 EDT 2013	1.0	
IBM Next 5 in 5 -- 2010.mp4	57278743	P8Admin	Fri Nov 01 05:58:20 EDT 2013	1.0	
Wearable PC.mp4	4471810	P8Admin	Fri Nov 01 05:59:05 EDT 2013	1.0	
2011 IBM 5 in 5 - Energy.mp4	20584712	P8Admin	Fri Nov 01 05:57:27 EDT 2013	1.0	
IBM Department of Energy Unveil Petaflop Supercomputer.mp4	8530957	P8Admin	Fri Nov 01 05:57:59 EDT 2013	1.0	
IBM Smarter Analytics converted.mp4	69266069	P8Admin	Fri Nov 01 05:58:39 EDT 2013	1.0	
Indeval IBM Smarter Planet Leadership Series Clien.mp4	14955141	P8Admin	Fri Nov 01 05:58:40 EDT 2013	1.0	
IBM Near Field Communication Worklight Demo.mp4	14870109	P8Admin	Fri Nov 01 05:58:16 EDT 2013	1.0	
IBM Research s nanoMRI microscope.mp4	4708902	P8Admin	Fri Nov 01 05:58:27 EDT 2013	1.0	

3/2/2014, 9:58 AM - 21 items were found.

The Breadcrumb module needs to set the parentFolder or searchTemplate parameters in the result set. When you click on the link, it will execute the open method of that item. The InlineMessage module shows messages of warning, info, confirm, and error with the setMessage() method.

```
var inlineMessageModule = this.navResult.getContentListModule( "inlineMessage" );
if (inlineMessageModule){
    inlineMessageModule.clearMessage();
    inlineMessageModule.setMessage("Result set items length is: "
+resultSet.items.length, "info");
}
```

The TotalCount module shows how many total results match the query criteria. To populate the TotalCount, you must set the **totalCount** and **totalCountType** properties of the result set. In IBM FileNet Content Manager V5.2 or above, there is an option that can be specified in a query string to return the result size by using the pageIterator.getTotalCount() method.

*select * from document where this infolder '\Test' OPTIONS (COUNT_LIMIT 1000)*

Note: Be aware that if the result size is less than the page size, then pageIterator.getTotalCount() will return null.

Congratulations! You've completed the lab.

End of Lab – Congratulations!