IBM Content Navigator
Version 2.0.3

Developing Applications with IBM Content Navigator APIs

IBM
IBM Content Navigator
Version 2.0.3

Developing Applications with IBM
Content Navigator APIs

IBM
Edition notices

This edition applies to version 8, release 5 of later of IBM Content Manager OnDemand for Multiplatforms (product number 5724-J33), version 9, release 0 or later of IBM Content Manager OnDemand for Multiplatforms (product number 5724-J33), version 8, release 5 or later of IBM Content Manager OnDemand for z/OS (product number 5697-N93), version 9, release 0 or later of IBM Content Manager OnDemand for z/OS (product number 5697-N93), version 7, release 1 or later of IBM Content Manager OnDemand for i (product number 5770-RD1), version 8, release 4 or later of IBM Content Manager Enterprise Edition (program number 5724-B19), version 8, release 4 or later of IBM Content Manager for z/OS (program number 5697-H60), version 5 release 0 or later IBM FileNet Content Manager (program number 5724-R81) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2012, 2014.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
ibm.com and related resources

Product support and documentation are available from ibm.com®.

Support and assistance

Product support is available on the web at:

IBM® Content Navigator
http://www.ibm.com/support/entry/portal/overview/software/Enterprise_Content_Manager/Content_Navigator

Information center

You can view the product documentation in an Eclipse-based information center on ibm.com. The information is available in the following information centers:

- IBM Content Manager Enterprise Edition Version 8.4 Information Center at
- IBM Content Manager OnDemand for Multiplatforms Version 8.5 Information Center at
- IBM FileNet® P8 Version 5.1 Information Center at
  http://publib.boulder.ibm.com/infocenter/p8docs/v5r1m0/index.jsp?topic=/com.ibm.developingeuc.doc/eucdi000.htm

PDF publications

You can view the PDF files online using the Adobe Acrobat Reader for your operating system. If you do not have the Acrobat Reader installed, you can download it from the Adobe Web site at http://www.adobe.com.

See the following PDF publication web site:

<table>
<thead>
<tr>
<th>Product</th>
<th>Web site</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Content Navigator</td>
<td><a href="http://www.ibm.com/support/docview.wss?uid=swg27025015">http://www.ibm.com/support/docview.wss?uid=swg27025015</a></td>
</tr>
</tbody>
</table>

“How to send your comments” on page vi
Your feedback helps IBM to provide quality information. Please share any comments that you have about this information or other documentation that IBM Software Development ships with its products.

“How Contacting IBM” on page vi
To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).
How to send your comments

Your feedback helps IBM to provide quality information. Please share any comments that you have about this information or other documentation that IBM Software Development ships with its products.

You can use any of the following methods to provide comments:

- Add comments by using the Comments pane at the bottom of every page in the information center. See Contributing to the information center.
- Send your comments by clicking the Feedback link at the bottom of any topic in the information center.
- Send your comments by using the online readers' comment form at [http://www.ibm.com/software/data/rcf/](http://www.ibm.com/software/data/rcf/).
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and publication number of the information (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

Consumability survey

Tell us how you feel about the value of quality content by taking the Importance of High Quality Technical Information survey at the following link: [http://www.ibm.com/survey/oid/wsb.dll/s/ag2c1](http://www.ibm.com/survey/oid/wsb.dll/s/ag2c1) If you want to help IBM make this product easier to install and use, take the Consumability Survey at the following link: [http://www.ibm.com/software/data/info/consumability-survey/](http://www.ibm.com/software/data/info/consumability-survey/)

Contacting IBM

To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

To learn about available service options, call one of the following numbers:

- In the United States: 1-888-426-4343
- In Canada: 1-800-465-9600

For more information about how to contact IBM, see the Contact IBM website at [http://www.ibm.com/contact/us/](http://www.ibm.com/contact/us/)

Parent topic: ibm.com and related resources” on page v

Related information:

- [Contributing to the information center](http://www.ibm.com/survey/oid/wsb.dll/s/ag2c1)

Share your opinions and contribute to the documentation in the information center. Simply sign in with your IBM ID to comment on the documentation, start discussion threads, share examples, rate topics, watch topics or threads, and more.
Developing applications with IBM Content Navigator

IBM Content Navigator provides a powerful platform for building custom web applications to manage content. You can use various extension points and application programming interfaces (APIs) to extend the existing web client by adding custom actions, menus, layouts, or services. You can also use the APIs to build custom applications that incorporate IBM Content Navigator features without using the standard web client.

You use the IBM Content Navigator Java™ API to create plug-ins to implement the functionality that you want to add to the web client. You use the IBM Content Navigator JavaScript API to create custom widgets to use with the plug-ins. In addition, you can use IBM Content Navigator to access the APIs for the content servers.

The sample code in the IBM Redbook, Customizing and Extending IBM Content Navigator, is based on the version 2.0.0 APIs, and requires changes to run on version 2.0.2.

What's new in version 2.0.3?
IBM Content Navigator Version 2.0.3 includes new customization options for application and plug-in development.

What's new for developers
Enhancements to the sample plug-in

Custom favorites feature
You can create a custom pane. For an example of how to use a custom pane, see the SampleFavoritesPane and SamplePluginFavoritesFeature APIs.

Responsive design and CSS enhancements
Design enhancements in IBM Content Navigator Version 2.0.3 give users more space on the screen for their current task, whether they are completing an entry template or editing a property dialog box. The style changes continue to use the same document object model (DOM) and cascading style sheets (CSS) as previous releases, so if you developed custom plug-ins or styles to override IBM Content Navigator styles, those customizations will not change. Customizations of banners, logos, colors, and themes can be modified from the administration tool so that you do not need to customize these items manually. You can simply import a theme or modify settings in the administration tool.

Feature icons are SVG files
The icons that come with IBM Content Navigator Version 2.0.3 are Scalable Vector Graphics (SVG) images. These icons are colored at run time.
according to the theme colors that you specify in your custom CSS. You can create your own SVG icons. See "Creating feature icons" on page 33.

Plug-in resources cached on the web browser
Resources in plug-ins are cached on the web browser. This change can improve the performance of the plug-ins.

New customization options

Get a list of dependencies for a particular plug-in
You can obtain a list of plug-ins that are dependent on a particular plug-in by using the getPluginDependencies method in the com.ibm.ecm.extension.Plugin.java class. This method returns an array of plug-in IDs.

Connecting to a data source by creating a repository type plug-in
You can create a plug-in for a custom repository type to authenticate, search, and provide result sets for repository servers. For example, you might want to use a custom repository type to extend IBM Content Navigator for your client application to use repositories that do not store documents. The custom repository type enables repository-specific authentication and other actions for your repository server.

Plug-ins can provide request and response filters on plug-in services
In previous versions of IBM Content Navigator, request and response filters were limited to services that are defined in IBM Content Navigator. In IBM Content Navigator Version 2.0.3, you can apply a filter to a plug-in provided service.

New JSON classes in com.ibm.ecm.json
The following JSON classes can be used to format data:
- JSONClassDefinitionResponse.java
- JSONContentClassesResponse.java
- JSONEditAttributesRequest.java
- JSONItemAttributesResponse.java
- JSONPrivilegesResponse.java

For more information, see com.ibm.ecm.json.

Entry templates are supported in the external data service (EDS) clientContext request parameter
You can include the active entry template's ID and name in the EDS context so that EDS implementations can return a different response if an entry template is active. For more information, see "POST method" on page 57 and ecm.model._HasAttributesMixin.

EDS is supported on IBM CMIS provided repositories
You can now create and use EDS services on IBM CMIS provided repositories that work with the IBM Content Navigator connector.
EDS can be used for the following actions in an IBM CMIS repository:
- addItem
- checkIn
- editAttributes
- openContentClass
- openItem
- getDependentAttributeInfo
Implementing dependent choice lists in search criteria by using EDS or IBM Content Manager foreign keys
EDS and IBM Content Manager foreign keys can be used to provide dependent choice lists in search criteria.

Authenticating to a nonrepository data source
You can use custom authentication when container-based authentication is not an option for your use case.

Social features
Social features include likes (recommendations), tags, number of downloads (download counts), and comments. Social features are enabled when you install the social add-on to an object store on FileNet P8 Version 5.2, and it will be enabled for all documents. For more information, see ecm.model.SocialItem and ecm.model.ContentItem.

FileNet P8 compound documents can be retrieved in result sets
You can retrieve FileNet P8 compound documents in IBM Content Navigator.

ACL services are supported for IBM CMIS for FileNet Content Manager and IBM CMIS for Content Manager
ACL services is a service that is provided in the OASIS Content Management Interoperability Services (CMIS) Version 1.0 OASIS Standard. You can use access ACL services to retrieve and modify security settings of individual objects (documents and folders) in the form of access control lists (ACLS). See http://docs.oasis-open.org/cmis/CMIS/v1.0/os/cmis-spec-v1.0.html#Toc243905485.

Enhancements for integrating with task manager
Task manager is a Java Platform, Enterprise Edition REST application that enables you to schedule and run tasks. For more information, see ecm.model.AsyncTaskSchedule and ecm.model.AsyncTaskAudit.

File tracker API
Provides the ability to access file tracker applet functions. File tracking streamlines the process of adding documents to and checking documents out of and in to the repository. It also enables administrators to manage how documents are stored on users’ computers. For more information, see ecm.model.FileTracker.

New methods in the ecm.model.Repository object for providing repository object values for search
Values that the ecm.model.SearchConfiguration object needs from the Repository object are available directly in the ecm.model.Repository object. The ecm.model.SearchConfiguration object no longer uses the ecm.model.admin.RepositoryConfig object.

New model object to create a search by using queries
Search now accepts JSON that defines a search query. The ecm.model.SearchQuery object provides the ability to search repositories by using a query string. The query string must conform to the query language specifications of the underlying repository.

New entry template designer widget
Users with the appropriate privileges can create entry
templates in IBM Content Navigator. For more information, see the ecm.widget.entryTemplateBuilder class.

Constructing URLs to customize what is displayed in the Work view
You can add parameters and values to a Work view URL to open and display a single in-basket, remove the tree view, and make other adjustments to how content is displayed when users access the Work view.

Constructing URLs to display the content list in a particular view mode
On a Favorites, Browse, Search, or Teamspace URL, you can add the defaultListView parameter to display the content list in a particular view mode: Details view, Magazine view, or Filmstrip view. If the specified view is not available on the desktop, the Details view is used as the default view.

Provide or override a custom search definition by using the searchCriteria parameter in a URL
You can pass custom search criteria to a saved search by using the searchCriteria URL parameter. You create and pass a JSON string array of property-value pairs to the URL. When the URL is invoked, users can modify the values in the search form, or you can additionally specify the runSearch=true parameter so that the URL invokes the search results directly.

Creating custom views by using the bookmark action handler
When you set up a custom view of a feature, for example, a custom teamspace, custom search pane, or custom document pane, the bookmark action handler ecm.widget.layout.BookmarkActionsHandler uses the custom features for folder and teamspace bookmarks if the custom features are set in the desktop.

Tip: The bookmark actions handler will use the first custom view that it finds for a custom feature. For example, if you define two different custom teamspace features in the same desktop, only the first one that is listed in the feature list will be used for bookmarks. You can override the entire bookmark layout per desktop. For an example of how to use a custom pane, see the sample favorites feature in the sample plug-in.

Providing a custom thumbnail generator
You can provide your own thumbnail generator. The function that you provide should return a base-64 encoded image to use in the HTML IMG element. For more information, see the addCustomThumbnailGenerator function. If you do not provide a custom thumbnail generator, the default thumbnail generator that is provided with IBM Content Navigator is used.

IDX SloshBucket widget
The new idx.widget.SloshBucket widget in IBM Dojo Extension for OneUI (IDX) now uses gridx and MemoryStore. The idx.widget.SloshBucket widget replaces the ecm.widget.SloshBucket widget.

Associating folders with entry templates
New classes are available to support a new dialog box that enables administrators to associate folders with one or more entry templates. When users add an item to a folder with one entry template association, that entry template is used to add the item into that folder. For more information, see ecm.model.EntryTemplateFolderAssociation.
Entry templates support IBM Content Manager object stores
Users can now add documents and folders to IBM Content Manager by using an entry template. For more information, see the ecm.model.EntryTemplate class.

Customizing the ContentView widget title
You can customize the text that is displayed to the left of the toolbar in the ContentView widget. For more information, see ecm.widget.viewer.ContentViewer.contentViewerTitle.

Mapping viewers to display previews according to MIME type
An isPreviewViewer method is added to the viewer definitions. When the following viewers are flagged as being a preview viewer by returning true for the isPreviewViewer method, when the preview action is invoked, a viewer map lookup is run and the matching viewer is used to display the preview:
- ICC conversion viewer (To use the ICC conversion viewer, you must install it and map it in a custom viewer map.)
- HTML conversion viewer
- Adobe Reader viewer
- PDF conversion viewer
- Daeja ViewONE Professional viewer
The HTML conversion viewer will remain the default viewer for the preview if no other viewers match the MIME type. For more information, see ecm.model.Desktop.getPreviewersForItem.

Added support to Atom (REST) bindings for IBM CMIS, which enables documents to be created with unique file names for the content stream (contentStreamFilename)
This enhancement enables implementations that are using either the setContentStream method or the appendContentStream method to create documents with unique content stream file names (contentStreamFilename) in the content-disposition header or Atom Publishing Protocol Slug header.

IBM CMIS users are able to create a document with unique values for the cmis:contentStreamFileName property and the chemistry:filename property.

Creating toolbars by instantiating toolbar objects
You can instantiate ecm.widget.Toolbar objects directly. You do not need to create a ContentList widget to create a toolbar.

Overriding the entry template destination ID when adding a document to the repository
You can pass in a destination folder for adding a document. You can specify for this value to override the destination that is defined by the entry template definition.

Deprecated API classes and methods
API changes in IBM Content Navigator Version 2.0.3 are backward-compatible with the IBM Content Navigator APIs that were available in version 2.0.1.

The following methods and classes in the Java APIs are deprecated starting in version 2.0.3:
### Table 1. Deprecated IBM Content Navigator Java classes and methods

<table>
<thead>
<tr>
<th>Deprecated classes and methods</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Deprecated as of version 2.0.2.1. This method is being removed because it has a dependency on the internal IBM Content Navigator CMObject class. Use the getCMAnnotationsXDO method, which returns an instance of dkXDO instead.</td>
</tr>
<tr>
<td>getCMAnnotations(CMItem, String, int) method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Deprecated as of version 2.0.2.1. This method is being removed because it has a dependency on the internal IBM Content Navigator CMObject class. Use the getCMBookmarksXDO method, which returns an instance of dkXDO instead.</td>
</tr>
<tr>
<td>getCMBookmarks(CMItem, String) method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Deprecated as of version 2.0.2.1. This method is being removed because it has a dependency on the internal IBM Content Navigator CMItem class. Use the getCMDocumentDDO method, which returns an instance of dkDDO instead.</td>
</tr>
<tr>
<td>getCMDocument(String, String, DKRetrieveOptionsICM) method</td>
<td></td>
</tr>
</tbody>
</table>

The following methods and classes in the JavaScript APIs are deprecated starting in version 2.0.3:

### Table 2. IBM Content Navigator JavaScript classes and methods that are deprecated starting in Version 2.0.3

<table>
<thead>
<tr>
<th>Deprecated methods and fields</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecm.widget.SloshBucket</td>
<td>Deprecated as of version 2.0.3. Use the idx.widget.SloshBucket class instead.</td>
</tr>
<tr>
<td>ecm.model.ContentItem.onCommentChanged</td>
<td>Deprecated as of version 2.0.3. Use the ecm.model.ContentItem.onCollaborate function instead.</td>
</tr>
<tr>
<td>ecm.model.ContentItem.modifyPermissions</td>
<td>Deprecated as of version 2.0.3. Use the Item model privModifyPermissions instead.</td>
</tr>
<tr>
<td>ecm.widget.layout.MainLayout.getBanner</td>
<td>Deprecated as of version 2.0.3. Use direct access to the banner.</td>
</tr>
<tr>
<td>ecm.widget.layout.MainLayout.getGlobalToolbar</td>
<td>Deprecated as of version 2.0.3. Use direct access to the global toolbar instead.</td>
</tr>
<tr>
<td>ecm.extension.PluginAction.getIcon</td>
<td>Deprecated as of version 2.0.3. Plug-ins should provide an icon cascading style sheet class and override getIconClass to specify images for buttons and menu items. See the ToolbarSample.html sample for an example.</td>
</tr>
<tr>
<td>ecm.model.Comment.deleteComment(callback, errorback)</td>
<td>Deprecated as of version 2.0.3. Use the ecm.model.SocialItem.deleteThis method instead.</td>
</tr>
<tr>
<td>ecm.model.Comment.isCommentDeletable</td>
<td>Deprecated as of version 2.0.3. Use the ecm.model.SocialItem.isDeletable method instead.</td>
</tr>
<tr>
<td>ecm.model.Comment.isCommentEditable</td>
<td>Deprecated as of version 2.0.3. Use the ecm.model.SocialItem.isEditable method instead.</td>
</tr>
<tr>
<td>ecm.model.Comment.isMyComment</td>
<td>Deprecated as of version 2.0.3. Use the ecm.model.SocialItem.isMine method instead.</td>
</tr>
<tr>
<td>ecm.model.Comment.saveComment</td>
<td>Deprecated as of version 2.0.3. Use the ecm.model.SocialItem.save method instead.</td>
</tr>
<tr>
<td>ecm.model.ContentItem.modifyPermissions</td>
<td>Deprecated as of version 2.0.3. Use the ecm.model.Item.privModifyPermissions method instead.</td>
</tr>
<tr>
<td>ecm.model.ContentItem.onCommentChanged</td>
<td>Deprecated as of version 2.0.3. Use the ecm.model.ContentItem.onCollaborate event instead.</td>
</tr>
<tr>
<td>ecm.model._HasAttributesMixin.</td>
<td>Deprecated as of version 2.0.3. Use retrieveAttributeDefinitions(callback, isBackgroundRequest, onError), retrieveAttributeDefinitionsForSearches(callback, includeSubclassDefinitions, onError), or retrieveDependentAttrDefs(params) instead.</td>
</tr>
<tr>
<td>retrieveDependentAttributeDefinitions(attributes, childComponentValues, callback, isBackgroundRequest, onError)</td>
<td></td>
</tr>
<tr>
<td>ecm.widget.AddContentItemPropertiesPane.getFolderName</td>
<td>Deprecated as of version 2.0.3. Use the getItemName method instead.</td>
</tr>
<tr>
<td>ecm.widget.GlobalToolbar</td>
<td>Deprecated as of version 2.0.3. Use the ecm.widget.Toolbar class instead.</td>
</tr>
</tbody>
</table>
Table 2. IBM Content Navigator JavaScript classes and methods that are deprecated starting in Version 2.0.3 (continued)

<table>
<thead>
<tr>
<th>Deprecated methods and fields</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ecm.widget.layout.MainLayout.getBanner</code></td>
<td>Deprecated as of version 2.0.3. Use direct access to the banner instead.</td>
</tr>
<tr>
<td><code>ecm.widget.layout.MainLayout.getGlobalToolbar</code></td>
<td>Deprecated as of version 2.0.3. Use direct access to the global toolbar instead.</td>
</tr>
<tr>
<td><code>ecm.widget.layout.NavigatorMainLayout.openNewSearch(item, data)</code></td>
<td>Deprecated as of version 2.0.3. Use the <code>openSearch</code> function and set <code>data.openNewTab</code> value to true.</td>
</tr>
<tr>
<td><code>ecm.widget.listView.modules.P8WorkToolbar</code></td>
<td>Deprecated as of version 2.0.3. Use the <code>ecm.widget.listView.modules.P8InbasketToolbar</code> module instead.</td>
</tr>
<tr>
<td><code>ecm.widget.listView.modules.Toolbar</code></td>
<td>Deprecated as of version 2.0.3. Use <code>ecm.widget.listView.modules.Toolbar2</code> module instead.</td>
</tr>
</tbody>
</table>

Parent topic: “Developing applications with IBM Content Navigator” on page 1

What's new in version 2.0.2?

IBM Content Navigator, Version 2.0.2 includes new customization options for application and plug-in development.

What's new for developers

Loading plug-ins by specifying a path on a class name

Previously, the only option to load a plug-in for IBM Content Navigator was to load the JAR file, which meant that you needed to build the JAR each time that you made a change that you wanted to test in the IBM Content Navigator server environment. Now you have the option to load a plug-in by specifying a path on a class name so that you can see your changes in real time without having to build and load a JAR file each time.

Additions to the sample plug-in

The sample plug-in includes several new sample classes.

Loading debug versions of the code by using the debug parameter

You can use `?debug=true` to load uncompressed JavaScript to make tracebacks and other diagnostics more understandable, or to more easily step between IBM Content Navigator JavaScript logic and plug-in logic in a debugger.

You can provide the debug versions of CSS and JavaScript in a plug-in and use the `?debug=true` parameter in your URL to load the uncompressed code.


New customization options

New features in the content list

The content list contains the following new features:

- Filter capabilities in teamspaces list and teamspace templates list
- Filmstrip view
- Ability to add custom widgets for each field in the magazine view

Form-based authentication

You can use form-based login to log in directly to the container.
Custom open actions
You can provide custom open actions in a plug-in. See
Plugin.getOpenAction and PluginOpenAction.

EDS support for IBM Content Manager OnDemand search
You can use EDS to control fields in Content Manager OnDemand search
forms. For example, you can set a field to be read-only, set initial values,
and implement a choice list for a String field.

Authentication service for Content Manager OnDemand single sign-on (SSO)
You can provide an authentication service for IBM Content Manager
OnDemand SSO. See Plugin.getODAuthenticationService.

Copyright information for a plug-in
You can provide copyright information for a plug-in that will be displayed
on an IBM Content Navigator About dialog. See Plugin.getCopyright.

Plug-in logging integration
You can add logging to plug-ins that integrates with IBM Content
Navigator logging. See PluginLogger and
PluginServiceCallbacks.getLogger.

VWSession object access from a plug-in
You can access the underlying VWSession object that is used by IBM
Content Navigator within a plug-in. See
PluginServiceCallbacks.getVWSession.

Per-user information
You can store per-user preference information from a plug-in. See
PluginServiceCallbacks.saveUserConfiguration and
PluginServiceCallbacks.loadUserConfiguration.

Access to mappings for labels of system properties
You can access IBM Content Navigator's mappings for labels of system
properties from within a plug-in. See
PluginServiceCallbacks.getSystemPropertyLabels.

Important: Synchronous module loading will be deprecated in Dojo 2.0. To
accommodate asynchronous module definition (AMD) loading, ecm.model.Desktop
methods, getActionHandler and getLayout, have been enhanced in version 2.0.2 to
accept callback functions. Callers using these ecm.model.Desktop methods to
retrieve widgets that might be dynamically loaded, should be changed to provide
the new callback function.

Deprecated API classes and methods

API changes in IBM Content Navigator, Version 2.0.2, are backwards-compatible
with the IBM Content Navigator APIs that were available starting in Version 2.0.1.

The following methods and classes in the Java APIs are deprecated starting in
Version 2.0.2 or Version 2.0.1:

<table>
<thead>
<tr>
<th>Deprecated classes and methods</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.ecm.extension.PluginServiceUtil class</td>
<td>This class is for internal use only and should not be used by plug-in writers.</td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.getCMAnnotations (CMItem, String) method</td>
<td>This method is being removed because it has a dependency on the internal IBM Content Navigator OIOObject class. Use the getCMAnnotations method, which returns an instance of dkXDO instead.</td>
</tr>
</tbody>
</table>
Table 3. Deprecated IBM Content Navigator Java classes and methods (continued)

<table>
<thead>
<tr>
<th>Deprecated classes and methods</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Deprecated as of version 2.0.2. This method is being removed because it has a dependency on the internal IBM Content Navigator CMObject class. Use getCMAnnotations, which returns an instance of dkXDO instead.</td>
</tr>
<tr>
<td>getCMAnnotations(CMItem, String, int) method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Deprecated as of version 2.0.2. This method is being removed because it has a dependency on the internal IBM Content Navigator CMObject class. Use getCMBookmarks, which returns an instance of dkXDO instead.</td>
</tr>
<tr>
<td>getCMBookmarks(CMItem, String) method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Deprecated as of version 2.0.2. This method is being removed because it has a dependency on the internal IBM Content Navigator CMItem class. Use getCMDocument, which returns an instance of DKDDO instead.</td>
</tr>
<tr>
<td>getCMDocument(String, String, DKRetrieveOptionsICM) method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.configuration.MenuConfig.getListTypes method</td>
<td>This method is no longer used.</td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>This method is deprecated because it uses internal class P8Connection. Use the getP8Connection method instead.</td>
</tr>
<tr>
<td>getP8Connection Object method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Retrieves the IBM FileNet P8 domain for the given repository. This method retrieves all properties of the domain which could have an performance impact. Use getP8Domain(String repositoryId, PropertyFilter propertyFilter) for better performance.</td>
</tr>
<tr>
<td>getP8Domain method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Deprecated as of version 2.0.2. This method is being deprecated because it has a dependency on the IBM Content Navigator internal class BaseMediator. Use the retrieveJSONAnnotations method instead.</td>
</tr>
<tr>
<td>retrieveAnnotations method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.extension.PluginServiceCallbacks.</td>
<td>Deprecated as of version 2.0.2. The returned DocumentContent class has been refactored into the extension package. Use the retrieveDocumentContent method.</td>
</tr>
<tr>
<td>retrieveContent method</td>
<td></td>
</tr>
<tr>
<td>com.ibm.ecm.configuration.MenuConfig.setListTypes method</td>
<td>This method is no longer used.</td>
</tr>
</tbody>
</table>

The following methods and classes in the JavaScript APIs are deprecated starting in version 2.0.2 or version 2.0.1:

Table 4. IBM Content Navigator Java classes and methods that are deprecated starting in Version 2.0.2

<table>
<thead>
<tr>
<th>Deprecated methods and fields</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>• desktop</td>
<td></td>
</tr>
<tr>
<td>• userid</td>
<td>userid Was never used or set internally. Use userId instead.</td>
</tr>
<tr>
<td>• onAddTeamspace(repository)</td>
<td>onAddTeamspace(repository) Use Repository onAddTeamspace event.</td>
</tr>
<tr>
<td>• onAddTeamspaceTemplate(repository)</td>
<td>onAddTeamspaceTemplate(repository) Use Repository onAddTeamspaceTemplate event.</td>
</tr>
<tr>
<td>• onTeamspaceListChange(repository)</td>
<td>onTeamspaceListChange(repository) Use Repository onTeamspaceListChange event.</td>
</tr>
<tr>
<td>• onTeamspaceTemplateListChange(repository)</td>
<td>onTeamspaceTemplateListChange(repository) Use Repository onTeamspaceTemplateListChange event.</td>
</tr>
<tr>
<td>• onWorklistContainersRetrieved(repository)</td>
<td>onWorklistContainersRetrieved(repository) Use Repository onWorklistContainersRetrieved event.</td>
</tr>
<tr>
<td>ecm.model.Repository</td>
<td>rootItem Use the retrieveItem method with a path of / to retrieve the root item. This value is null until a retrieveItem with a path of / is performed.</td>
</tr>
<tr>
<td>• rootItem</td>
<td></td>
</tr>
<tr>
<td>Deprecated class and field</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>ecm.model.SearchCriterion</td>
<td>value</td>
</tr>
<tr>
<td>ecm.model.SearchTemplate</td>
<td>toRecentSearchJson(contextId, userId)</td>
</tr>
<tr>
<td>ecm.widget.Banner</td>
<td>setUser(repository)</td>
</tr>
<tr>
<td>ecm.widget.dialog.ErrorDialog</td>
<td>messageQueryUrl</td>
</tr>
<tr>
<td>ecm.widget.layout.CommonActionsHandler</td>
<td>actionRefreshAttachments(repository, items)</td>
</tr>
<tr>
<td>ecm.widget.listView.MagazineViewDecorator</td>
<td>contentCellDecoratorCM8(data, rowId, rowIndex), contentCellDecoratorCMIS(data, rowId, rowIndex), contentCellDecoratorP8(data, rowId, rowIndex)</td>
</tr>
<tr>
<td>ecm.widget.search.SearchBuilder</td>
<td>showDocumentInfoPane</td>
</tr>
<tr>
<td>ecm.widget.search.SearchDialog</td>
<td>builder</td>
</tr>
<tr>
<td>ecm.widget.search.SearchSelector</td>
<td>showRecents, getRepository</td>
</tr>
<tr>
<td>ecm.widget.search.SearchSelectorDialog</td>
<td>selectionMode</td>
</tr>
<tr>
<td>ecm.widget.search.SearchTab</td>
<td>showClear, showEdit, showingContentList, showingSearchToggleArea, resetPanes()</td>
</tr>
<tr>
<td>Deprecated methods and fields</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------</td>
</tr>
</tbody>
</table>
| ecm.widget.viewer.AjaxViewer  | buildPageFrame()  
  - buildPageFrame()  
  - buildToolbar()  
  - displayDocument(docUrl, docName, contentType, serverType)  
  - displayPage()  
  - getPageInfo(pageNo)  
  - onresize()  
  - retrievePageCount()  
  - retrieveViewServicesInfo()  
  - save()  
  - updatePageSelect()  
  - updateScaleFromSelection()  
  - updateToolStates() | Deprecated as of version 2.0.2. This method will be made private. |

| ecm.widget.viewer.ContentViewer | close(item)  
  - close(item) | As of release 2.0.2. This method is not implemented. |
<table>
<thead>
<tr>
<th>Deprecated methods and fields</th>
<th>Comments</th>
</tr>
</thead>
</table>
| ecm.widget.viewer.ContentViewerPane | closeItem()  
|                               | Deprecated as of version 2.0.1. Use the DocViewer framework for document viewer specific operations instead. |
|                               | getItemIndex()  
|                               | Deprecated as of version 2.0.1. This method is intended for internal use, and is being replaced. |
|                               | getNextItem()  
|                               | Deprecated as of version 2.0.1. Use the ViewerItem framework for item specific operations instead. |
|                               | getPrevItem()  
|                               | Deprecated as of version 2.0.1. Use the ViewerItem framework for item specific operations instead. |
|                               | getSaveOrWaitStatus(listHeading, nameList, asHtml)  
|                               | Deprecated as of version 2.0.1. There is no replacement. Intended for internal use only. |
|                               | getSaveOrWaitStatusMessage(nameList, asHtml)  
|                               | Deprecated as of version 2.0.1. There is no replacement. Intended for internal use only. |
|                               | hasNextItem()  
|                               | Deprecated as of version 2.0.1. Use the ViewerItem framework for item specific operations instead. |
|                               | hasPrevItem()  
|                               | Deprecated as of version 2.0.1. Use the ViewerItem framework for item specific operations instead. |
|                               | isDirty()  
|                               | Deprecated as of version 2.0.1. Use the DocViewer framework for document viewer specific operations instead. |
|                               | isLocked()  
|                               | Deprecated as of version 2.0.1. Use the DocViewer framework for document viewer specific operations instead. |
|                               | isPrinting()  
|                               | Deprecated as of version 2.0.1. Use the DocViewer framework for document viewer specific operations instead. |
|                               | isViewerLoaded()  
|                               | Deprecated as of version 2.0.1. Use docViewer.isLoaded() instead. |
|                               | nextPageRequired()  
|                               | Deprecated as of version 2.0.1. Use the ViewerItem framework for item specific operations instead. |
|                               | onClose(closeCallback, cancelCallback)  
|                               | Deprecated as of version 2.0.1. Replaced by the closeViewer event. |
|                               | retrieveNextPage(callback)  
|                               | Deprecated as of version 2.0.1. Use the ViewerItem framework for item specific operations instead. |
|                               | unloadIframe()  
<p>|                               | Deprecated as of version 2.0.1. Use the DocViewer framework for document viewer specific operations instead. |</p>
<table>
<thead>
<tr>
<th>Deprecated methods and fields</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecm.widget.viewer.ContentViewerReLogin widget</td>
<td>This widget is deprecated starting in version 2.0.1. This widget is not intended for reuse, and there is no replacement.</td>
</tr>
<tr>
<td>ecm.widget.viewer.model.ViewerItem</td>
<td></td>
</tr>
<tr>
<td>• getId()</td>
<td>Deprecated as of version 2.0.2. Accessors are being replaced with accessing public properties directly. Use viewerItem.item.id.</td>
</tr>
<tr>
<td>• getItem()</td>
<td>Deprecated as of version 2.0.2. Accessors are being replaced with accessing public properties directly. Use viewerItem.item.</td>
</tr>
<tr>
<td>• getRepository()</td>
<td>Deprecated as of version 2.0.2. Accessors are being replaced with accessing public properties directly. Use viewerItem.item.repository.</td>
</tr>
<tr>
<td>• getResultSet()</td>
<td>Deprecated as of version 2.0.2. Accessors are being replaced with accessing public properties directly. Use viewerItem.item.resultSet.</td>
</tr>
<tr>
<td>ecm.widget.viewer.OutsideInViewerMixin widget</td>
<td>This widget is deprecated starting in version 2.0.2. This class contains only private methods, and will be made private.</td>
</tr>
</tbody>
</table>

**Parent topic:** "Developing applications with IBM Content Navigator” on page 1
IBM Content Navigator development architecture

IBM Content Navigator uses a model-view-controller (MVC) architecture.

As the following diagram shows, the components of IBM Content Navigator are separated functionally according to the MVC design:

**Web client**

The web client acts as a controller that provides the communication between the view and the model. Users interact with the widgets that are contained in the web client. The web client translates these interactions into the actions that are performed by instances of the model classes.

**Visual widget library**

This library of JavaScript classes defines widgets that are used to build the view for IBM Content Navigator.

**Modeling library**

This library of JavaScript classes defines the model for IBM Content Navigator. The modeling classes provide the data, server interaction and caching. These classes define communication with the midtier services to provide access to the content repositories and their respective application programming interfaces (APIs).

You can use the modeling classes to customize the communication and data exchange between the web client and the repositories.
You can customize IBM Content Navigator at each level of the architecture by using custom plug-ins. You can create plug-ins to implement custom widgets, services, and other components. By creating custom plug-ins, you can integrate information and content from external sources and augment or modify the behavior of IBM Content Navigator.

IBM Content Navigator uses plug-ins to implement various features. For example, IBM Watson Content Analytics is integrated as a plug-in to IBM Content Navigator. The external data service plug-in augments the behavior of many IBM Content Navigator widgets to support the integration of data from sources other than the content repositories. You can use this feature to add such features as choice lists, choice list dependencies, and validation to the web client.
IBM Content Navigator midtier services

The IBM Content Navigator services provide the connections to the repositories. In addition, the services support features such as search and document viewing across the repositories.

IBM Content Navigator includes services to connect to:

- IBM Content Manager Express® Edition
- IBM Content Manager for z/OS® repositories
- IBM Content Manager Enterprise Edition repositories
- IBM Content Manager OnDemand repositories
- IBM FileNet Content Manager repositories
- Content Management Interoperability Services (CMIS) repositories

IBM Content Navigator uses the application programming interfaces (APIs) of each repository type to establish connections to these repositories. You can create custom plug-ins that use these APIs to perform actions on repository items.

IBM Content Navigator also includes plug-ins to support the following services:

**IBM Watson Content Analytics plug-in**

Supports searching across different repositories and data sources that are configured for use with IBM Content Navigator. You can search with simple or advanced queries. In addition, you can use facets to organize and classify items, which makes it easier to find documents quickly. The plug-in provides actions that you can use to work with the search results. For example, you can edit documents that are returned by the search or add the documents to workflows.

**External data service plug-in**

Supports integration of data from an external data source into IBM Content Navigator. You can use this external data to customize field properties and manage property behavior in IBM Content Navigator.

To use this plug-in, you must create a service to pass the data from the external data source to IBM Content Navigator.
IBM Content Collector viewer plug-in
Supports the use of IBM Content Collector to view archived email, IBM Lotus® Notes® documents, and IBM Connections archives.

To use this plug-in, you must:
- Provide the URL to your IBM Content Collector web service
- Update the IBM Content Navigator viewer mappings to use the IBM Content Collector viewer for the IBM Content Collector archive content types

Parent topic: IBM Content Navigator development architecture on page 15
Related tasks:
- Creating an external data service for IBM Content Navigator on page 50
  Use the EDS REST protocol to create an external data service that specifies requests to get data from an external data source to customize field properties and manage property behavior in IBM Content Navigator. You implement an external data service as a web application.
- Creating a service on page 40
  You can create a plug-in to define a service that performs operations on a content server. For example, you can create a service that provides a viewer for specific document types on a IBM FileNet P8 server.
Samples for IBM Content Navigator

The IBM Content Navigator software package includes several samples: a plug-in, the external data service plug-in, and a set of web pages. You can use these samples to create custom applications.

**Sample external data service**

The sample external data service (EDS) implements a service that uses various data from an external source. For example, this sample includes the implementations of the following EDS capabilities:

- Simple validation using regular expressions
- Choice lists
- Dependent choice lists
- Workflow-specific choice lists

The sample web pages are available in the `ECMClient_installdir\samples\sampleEDSService` directory.

**Sample web pages**

This sample web application provides HTML pages that use IBM Content Navigator modeling classes and widget classes. The pages demonstrate different aspects of the IBM Content Navigator toolkit.

The sample web pages are available in the `ECMClient_installdir\samples\samplePages` directory.

**Sample plug-in application**

The sample plug-in implements a plug-in that demonstrates several extension capabilities available to plug-in writers. The files in this sample application show you how to create custom components such as a custom viewer, menus, features, and services for use in the IBM Content Navigator web client.

The sample plug-in also includes code that shows you how to use action handling for custom file uploads. This functionality in the sample includes client single file upload control, server access of the uploaded document in the plug-in action, and sample file manipulation code.

The files for the sample plug-in are available in the `ECMClient_installdir\samples\samplePlugin` directory. The `SamplePlugin.jar` file that you can deploy by using the IBM Content Navigator administration tool is available in the `ECMClient_installdir\navigator` directory.
Table 5. Sample files in the sample plug-in application:

<table>
<thead>
<tr>
<th>Task</th>
<th>Files and description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a plug-in</td>
<td>The following files must be used to instantiate plug-ins and methods:</td>
</tr>
<tr>
<td></td>
<td>- SamplePlugin.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePlugin.css</td>
</tr>
<tr>
<td></td>
<td>- SamplePlugin.js</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginAction.gif</td>
</tr>
<tr>
<td></td>
<td>- ConfigurationPane.html</td>
</tr>
<tr>
<td></td>
<td>- ConfigurationPane.js</td>
</tr>
<tr>
<td>Create a custom menu action</td>
<td>The following sample files can be used to create a custom menu action:</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginAction.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginCheckInAction.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginFilteredAction.java</td>
</tr>
<tr>
<td></td>
<td>- CustomAction.js</td>
</tr>
<tr>
<td></td>
<td>- FileUploadCustomAction.js</td>
</tr>
<tr>
<td></td>
<td>- FileUploadPopupDialog.js</td>
</tr>
<tr>
<td>Create a custom menu</td>
<td>The following sample files can be used to create a custom menu:</td>
</tr>
<tr>
<td>feature</td>
<td>- SamplePluginContextMenuMenuType.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginContextMenuMenu.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginToolbarMenuType.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginToolbarMenu.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginToolbarMenuType2.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginToolbarMenu2.java</td>
</tr>
<tr>
<td>Create a custom feature</td>
<td>The following sample files can be used to create a custom feature:</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginFeature.java</td>
</tr>
<tr>
<td></td>
<td>- SampleFeaturePane.js</td>
</tr>
<tr>
<td>Create a custom layout</td>
<td>The following sample files can be used to create a custom feature:</td>
</tr>
<tr>
<td></td>
<td>- SampleLayout.js</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginLayout.java</td>
</tr>
<tr>
<td>Create a custom repository</td>
<td>The following sample files can be used to create a custom repository type:</td>
</tr>
<tr>
<td>type</td>
<td>- SamplePluginRepositoryType.java</td>
</tr>
<tr>
<td>Create a request or</td>
<td>The following sample files can be used to create a request or response filter:</td>
</tr>
<tr>
<td>response filter</td>
<td>- SamplePluginResponseFilter.java</td>
</tr>
<tr>
<td>Create a custom service</td>
<td>The following sample files can be used to create a custom service:</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginService.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginViewerDef.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginViewerService.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginServiceCM.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginServiceCMIS.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginServiceOD.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginGetAnnotationsService.java</td>
</tr>
<tr>
<td></td>
<td>- SamplePluginGetContentService.java</td>
</tr>
</tbody>
</table>
### Table 5. Sample files in the sample plug-in application: (continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Files and description</th>
</tr>
</thead>
</table>
| Create a custom service that uploads files | The following sample files can be used to create a custom service that uploads files:  
  - SamplePluginFileUploadAction.java  
  - SamplePluginFileUploadService.java  
  - FileUploadPopupDialog.html |
| Create a custom viewer | The following sample files can be used to create a custom viewer:  
  - SampleImageViewer.js  
  - SamplePluginViewerDef.java  
  - SamplePluginViewerService.java  
  - SampleImageViewer.html |
| Create a custom widget | The following sample files can be used to create a custom widget:  
  - MessagesDialog.js  
  - MessagesDialog.html  
  - PopupDialog.js  
  - PopupDialog.html  
  - SamplePlugin.js |
| Create a custom section in the properties pane | SampleItemPropertiesPaneExtension.js |

**Related concepts:**

- [“Sample external data service” on page 70](#)

In the sample external data service, the values for a choice list are used from an external data source.

**Related tasks:**

- [“Creating plug-ins to IBM Content Navigator” on page 25](#)

You can create custom plug-ins to add features such as menus and services to IBM Content Navigator.
Constructing a URL for IBM Content Navigator

IBM Content Navigator desktops, features, folders, and documents can be started or opened remotely by sending the user’s browser to a specified URL that includes the item’s unique identifier. This capability makes referencing documents and folders in a custom application simple and consistent, even when the source code in IBM Content Navigator is changed.

Procedure

To construct a URL for IBM Content Navigator desktops, features, documents, or folders:

1. Create the base URL as described in Constructing a URL for IBM Content Navigator
2. Optional: For a URL to a custom search form or search results, create the URL as described in Creating a URL to a custom search form or search results
3. Optional: For a shortcut URL to a worklist or in-basket, create the URL as described in Creating shortcut URLs to worklists and in-baskets
Creating plug-ins to IBM Content Navigator

You can create custom plug-ins to add features such as menus and services to IBM Content Navigator.

Before you begin

You must understand the following programming languages to create a plug-in to IBM Content Navigator:

- Java
- JavaScript

In addition, you must understand web programming to implement the JavaScript functions for the actions that are defined by the plug-in.

You should also be familiar with the Dojo Toolkit, which is the toolkit that IBM Content Navigator uses.

About this task

The plug-in that you create to use with IBM Content Navigator must provide:

- The web browser logic that enables users to call the plug-in
- The midtier server logic that enables the plug-in to load the plug-in components.

If administrators must provide additional information for the plug-in to function, you must also provide a configuration component. For example, the IBM Watson Content Analytics plug-in requires administrators to provide a URL to the IBM Watson Content Analytics server. Administrators use the IBM Content Navigator administration tool to configure the plug-in.

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

The web browser logic

The web browser logic is a required component of the plug-in that enables users to call the plug-in from the web client.

The web-browser logic component of your plug-in communicates with the IBM Content Navigator client. The web client is built by using the IBM Content Navigator visual widgets, which are based on the Dojo Toolkit. You should use the IBM Content NavigatorC visual widgets and the Dojo Toolkit to build your plug-in.

The web browser logic is implemented in JavaScript.

The midtier server logic

The midtier server logic is a required component of the plug-in. This logic calls the APIs for the content servers to retrieve the data that is used by the plug-in or that is displayed to users.

The midtier logic component of your plug-in communicates with the IBM Content Navigator services component on the web application server.

The midtier server logic is implemented in Java.
The configuration component

The configuration component is an optional component of the plug-in that enables an instance of the plug-in to be configured with the IBM Content Navigator administration tool. See step 2 in the following procedure for information about creating a configuration component for your plug-in.

The configuration component is implemented in Java.

Procedure

To create a plug-in to IBM Content Navigator:

1. Extend the Plugin class to create the plug-in.
2. If the plug-in requires configuration, define a configuration pane to display in the IBM Content Navigator administration tool:
   a. Extend the PluginConfigurationPane.js class to define parameters that need to be configured.
   b. Create an HTML file to define a form template to define the user interface for the configuration pane.
   c. In your Plugin class file, implement the getConfigurationDijitClass method to return the PluginConfigurationPane.js class.
3. Extend the appropriate ecm.extension classes to create the components that define the functionality to be provided by the plug-in:

<table>
<thead>
<tr>
<th>Component type</th>
<th>ecm.extension class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu action</td>
<td>PluginAction</td>
</tr>
<tr>
<td>Menu</td>
<td>PluginMenuType and PluginMenu</td>
</tr>
<tr>
<td>Feature</td>
<td>PluginFeature</td>
</tr>
<tr>
<td>Layout</td>
<td>PluginLayout</td>
</tr>
<tr>
<td>Request filter</td>
<td>PluginRequestFilter</td>
</tr>
<tr>
<td>Response filter</td>
<td>PluginResponseFilter</td>
</tr>
<tr>
<td>Service</td>
<td>PluginService</td>
</tr>
<tr>
<td>Viewer</td>
<td>PluginViewerDef</td>
</tr>
<tr>
<td>Widget</td>
<td>Not applicable. Instead, you extend the Dojo dijit._widget class or another Dojo dijit class.</td>
</tr>
</tbody>
</table>

4. In your Plugin class file, implement the appropriate get methods to identify the custom components that are included in this plug-in:

<table>
<thead>
<tr>
<th>Component type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu action</td>
<td>getActions</td>
</tr>
<tr>
<td>Menu</td>
<td>getMenuTypes and getMenus</td>
</tr>
<tr>
<td>Feature</td>
<td>getFeatures</td>
</tr>
<tr>
<td>Layout</td>
<td>getLayouts</td>
</tr>
<tr>
<td>Request filter</td>
<td>getRequestFilters</td>
</tr>
<tr>
<td>Response filter</td>
<td>getResponseFilters</td>
</tr>
<tr>
<td>Service</td>
<td>getServices</td>
</tr>
<tr>
<td>Viewer</td>
<td>getViewers</td>
</tr>
<tr>
<td>Widget</td>
<td>getScript</td>
</tr>
</tbody>
</table>
5. Package the plug-in files in a JAR file.
6. Deploy the JAR file into your web application server.

**Important:** The plug-in JAR file must be available on a URL addressable web application server or the plug-in will not work in the web client.

7. Use the IBM Content Navigator administration desktop to add the new plug-in.

**Sample plug-in definition and configuration files**

The following files in the sample plug-in define the plug-in and the configuration:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePlugin.java</td>
<td>This file extends the Plugin class to list the custom components that are provided by the sample plug-in.</td>
</tr>
<tr>
<td>SamplePlugin.js</td>
<td>This file extends the Plugin class to list the custom components that are provided by the sample plug-in.</td>
</tr>
<tr>
<td>ConfigurationPane.js</td>
<td>This file defines the configuration pane that is used for the sample plug-in.</td>
</tr>
<tr>
<td>ConfigurationPane.html</td>
<td>This file provides the HTML form template that is used for the sample plug-in configuration pane.</td>
</tr>
</tbody>
</table>

"Creating the plug-in components" on page 28
You can create plug-ins to add components such as menus, services, widgets, and layouts to IBM Content Navigator.

"Packaging the plug-in components" on page 44
You extend the Plugin class to identify the components that are used for a custom plug-in. You then create a JAR file to package the components deploy to IBM Content Navigator.

**Related concepts:**
"IBM Content Navigator development architecture" on page 15
IBM Content Navigator uses a model-view-controller (MVC) architecture.

**Related reference:**
"Samples for IBM Content Navigator" on page 19
The IBM Content Navigator software package includes several samples: a plug-in, the external data service plug-in, and a set of web pages. You can use these samples to create custom applications.

**Related information:**
- Dojo Toolkit Reference Guide
  Find information about the Dojo Toolkit that is used by IBM Content Navigator.
- IBM Redbooks: Customizing and Extending IBM Content Navigator
  Find more information about customizing and extending IBM Content Navigator.
- Class ecm.widget.admin.PluginConfigurationPane
  Provides a base class that can be extended to create a configuration interface for a plug-in.
- Class Plugin
  Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.
Creating the plug-in components

You can create plug-ins to add components such as menus, services, widgets, and layouts to IBM Content Navigator.

**Creating a menu action** on page 29
You can create a plug-in to provide an action to use in IBM Content Navigator. For example, you might define a custom action that imports documents or that generates a custom report generation for a set of selected documents. This action can be added to toolbars and context menus.

**Creating a menu** on page 30
You can create a plug-in to provide a custom toolbar or context menu to use in IBM Content Navigator.

**Creating a feature** on page 32
You can create a plug-in to add a major feature to IBM Content Navigator. For example, you might create a feature to show statistical data on content usage or a summary of recent document activity. You might also create a feature or to select documents for archival. This feature can be displayed with the features that are provided by IBM Content Navigator such as favorites, browse, search, and administration.

**Creating feature icons** on page 33
The feature icons that are used in IBM Content Navigator are Scalable Vector Graphics (SVG) images. These icons can be recolored at run time by using colors that are specified in your theme CSS. You can create your own SVG icons.

**Creating a layout** on page 35
You can create a plug-in to provide a custom layout to use in IBM Content Navigator. The layout defines the arrangement of the widgets that are used in a desktop.

**Creating a repository type** on page 36
You can create a plug-in for a custom repository type to authenticate, search, and provide result sets for repository servers. For example, you might want to use a custom repository type to extend IBM Content Navigator for your client application to use repositories that do not store documents. The custom repository type enables repository-specific authentication and other actions for your repository server.

**Creating a request or response filter** on page 38
You can create a plug-in to filter a request that is made to a service or to filter a response that is received from a service. For example, you might create a request filter to modify the parameters that are sent to a service to provide custom validations and error handling. You can also create a request filter to override a service. For example, you might create a response filter to apply custom formatting to the data that is received from a service before the data is displayed.

**Creating a service** on page 40
You can create a plug-in to define a service that performs operations on a content server. For example, you can create a service that provides a viewer for specific document types on a IBM FileNet P8 server.

**Creating a viewer** on page 41
You can create a plug-in to provide a custom viewer. Administrators can then map the custom viewer to document types by using the IBM Content Navigator administration tool.

**Creating a widget** on page 43
You can create a widget to provide the user interface for your plug-in. For
example, you might create a widget that enables users to run a custom action. You also might create a widget to enable administrators to configure the plug-in.

Creating a menu action

You can create a plug-in to provide an action to use in IBM Content Navigator. For example, you might define a custom action that imports documents or that generates a custom report generation for a set of selected documents. This action can be added to toolbars and context menus.

About this task

Procedure

To create an action:

1. Implement a JavaScript function to define the behavior of the action. For information about actions in IBM Content Navigator, see the ecm.model.Action.js class.
2. If you want to specify when the action is enabled or disabled, extend the ecm.model.Action.js class.
3. Extend the PluginAction.java class to define the plug-in for the action:
   a. Implement the getActionFunction method to return name of the JavaScript function that you created in step 1.
   b. If this action overrides the standard code for enabling actions, implement the getActionModelClass method to return name of the JavaScript class that you created in step 2.
4. Add the action as a component in your Plugin.java subclass. To add the action, implement the getFeatures method to return an instance of the PluginAction.java class that you created in step 3.

Sample plug-in files for a custom menu action

The following files in the sample plug-in define a custom menu action:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginAction.java</td>
<td>This file extends the PluginAction.java class to define a sample action.</td>
</tr>
<tr>
<td></td>
<td>The behavior of this action is defined by the samplePluginFunction that is defined in the SamplePlugin.js file.</td>
</tr>
<tr>
<td>SamplePluginCheckInAction.java</td>
<td>This file extends the PluginAction.java class to define a sample action.</td>
</tr>
<tr>
<td></td>
<td>The behavior of this action is defined by the samplePluginFunction that is defined in the SamplePlugin.js file.</td>
</tr>
</tbody>
</table>
Table 7. Sample files that define a plug-in menu action (continued)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginFilteredAction.java</td>
<td>This file extends the PluginAction.java class to define a sample action. This class implements an action that is available only for documents with a MIME type of text. The behavior of this action is defined by the samplePluginFilteredAction function that is defined in the SamplePlugin.js file.</td>
</tr>
<tr>
<td>CustomAction.js</td>
<td>This file implements the isEnabled function so that an action is enabled only for documents with a MIME type of text.</td>
</tr>
</tbody>
</table>

What to do next

Add the action to a custom toolbar or context menu by implementing the getMenuItems method in a PluginMenu subclass to return an instance of this PluginAction class. This method adds the action to the custom toolbar or menu that is represented by the PluginMenu subclass.

After you add the plug-in in the web client, use the IBM Content Navigator administration tool to add the action to a toolbar or context menu.  

Parent topic: “Creating the plug-in components” on page 28  

Related information:

- **Class ecm.model.Action**  
  Represents a user-executable action. This action can be configured for a desktop in the IBM Content Navigator administration tool.

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

- **Class Plugin**  
  Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.

Creating a menu

You can create a plug-in to provide a custom toolbar or context menu to use in IBM Content Navigator.

About this task

To create a menu, you must first create the menu type. The menu type indicates where and how the menu will appear in the user interface. For example, you might define a menu type to create a context menu for a custom widget. You can use the menu type for multiple menus.
Procedure

To create a custom menu:

1. Extend the `PluginMenuType.java` class to define the menu type:
   
   If the menu is a toolbar, implement the `isToolbar` method to return `true`. If you do not implement this method or if the method returns `false`, the menu is a context menu.

   For a default menu, the menu type provides an identifier for the default menu. When a user creates a copy of this menu, the menu type is identified in the copy to link it back to the original menu.

2. Extend the `PluginMenu.java` class to define the menu:
   
   a. Implement the `getId` method to return the identifier for the menu. Prefix the identifier with "Default" if users will be able to create copies of this menu.
   
   b. Implement the `getMenuType` method to return the `PluginMenuType` class that you created in step 1.
   
   c. Implement the `getMenuItems` method to return instances of the actions that are included in the menu.

3. Add the menu as a component in your `Plugin.java` subclass:
   
   a. Implement the `getMenuTypes` method to return an instance of the `PluginMenuType.java` subclass that you created in step 1.
   
   b. Implement the `getMenus` method to return an instance of the `PluginMenu.java` subclass that you created in step 2.

Sample plug-in files for a custom menu

The following files in the sample plug-in define a context menu and two toolbar menus:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginContextMenuMenuType.java</td>
<td>This file extends the <code>PluginMenuType.java</code> class to create a menu type for a context menu.</td>
</tr>
</tbody>
</table>
| SamplePluginContextMenuMenu.java | This file extends the `PluginMenu.java` class to defines a context menu that includes the following actions:  
  - A sample action, which is defined in the `SamplePluginAction.java` file  
  - Properties  
  - Help  
  - About  
  - Preview  
  - Open  

  The following actions are in the available actions box:  
  - Custom Checkin  
  - Only For Text Docs |
| SamplePluginToolbarMenuType.java | This file extends the `PluginMenuType.java` class to create a menu type for a toolbar. |
Table 8. Sample files that define a plug-in menu (continued)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginToolbarMenu.java</td>
<td>This file extends the PluginMenu.java class to defines a toolbar that includes the following actions:</td>
</tr>
<tr>
<td></td>
<td>• A sample action, which is defined in SamplePluginAction.java file</td>
</tr>
<tr>
<td></td>
<td>• Help</td>
</tr>
<tr>
<td></td>
<td>• About</td>
</tr>
<tr>
<td>SamplePluginToolbarMenuType2.java</td>
<td>This file extends the PluginMenuType.java class to create a menu type for a toolbar.</td>
</tr>
<tr>
<td>SamplePluginToolbarMenu2.java</td>
<td>This file extends the PluginMenu.java class to defines a toolbar that includes the following actions:</td>
</tr>
<tr>
<td></td>
<td>• A sample action, which is defined in the SamplePluginAction.java file</td>
</tr>
<tr>
<td></td>
<td>• Custom Checkin, which is defined in the SamplePluginCheckInAction.java file</td>
</tr>
<tr>
<td></td>
<td>• Help</td>
</tr>
</tbody>
</table>

What to do next

After you add the plug-in in the web client, use the IBM Content Navigator administration tool to add the menu to a desktop.

Parent topic: “Creating the plug-in components” on page 28

Related information:
- Class PluginMenuType
  Provides an abstract class that is extended to define a custom menu type provided by a plug-in.
- Class PluginMenu
  Provides an abstract class that is extended to define a menu in a plug-in.
- Class Plugin
  Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.

Creating a feature

You can create a plug-in to add a major feature to IBM Content Navigator. For example, you might create a feature to show statistical data on content usage or a summary of recent document activity. You might also create a feature or to select documents for archival. This feature can be displayed with the features that are provided by IBM Content Navigator such as favorites, browse, search, and administration.

Procedure

To define a feature:
1. Create an icon to represent the feature in the web client. The icon must be 24 pixels wide by 24 pixels high on a transparent background. Make the target dimensions fit within 22 pixels by 22 pixels and surround it with 1 pixel of
padding. For best results and maintainability, create the icon as a Scalable Vector Graphics (SVG) image. Alternatively, you can create the icon as a transparent PNG file.

2. Create a widget to provide the primary pane that defines the user interface for the feature. This widget provides users with access to the behavior of the feature.

3. Extend the PluginFeature class to describe the feature:
   a. Implement the getIconUrl method to return the class name of the icon that you created in step 1 on page 32.
   b. Implement the getSvgFilePath method to return the path to the SVG file, if you created one in step 1 on page 32.
   c. Implement the getContentType method to return an instance of the widget that you created in step 2.

4. Add the service as a component in your Plugin.java subclass. Implement the getFeatures method to return an instance of the PluginFeature class that you created in step 3.

Sample plug-in files for a custom feature

The following files in the sample plug-in define a feature:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginFeature.java</td>
<td>This file extends the PluginFeature class to create a sample feature.</td>
</tr>
</tbody>
</table>

What to do next

After you add the plug-in in the web client, use the IBM Content Navigator administration tool to add the feature to a desktop. You add the feature on the Appearance tab for the desktop.

Related information:

- **Class PluginFeature**
  Provides an abstract class that is extended to define a feature provided by the plug-in. 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.

- **Class Plugin**
  Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.

Creating feature icons

The feature icons that are used in IBM Content Navigator are Scalable Vector Graphics (SVG) images. These icons can be recolored at run time by using colors that are specified in your theme CSS. You can create your own SVG icons.
About this task

You create two versions of the same set of icons so that one set of icons is styled for when the feature is in the inactive, unselected state, and the other set of icons is styled for when the feature is in the active, selected state. You create and save the icons in a single file, for example, on a single canvas or artboard.

Procedure

To create SVG feature icons to represent features in the web client:

1. Open a vector drawing application, for example, Adobe Illustrator. Then, open a new canvas or artboard.
2. Create the icons in a single row on the artboard with the following dimensions:

   Follow IBM Design icon guidelines for best practices and to keep your icons consistent:

   - Icons must be 24 pixels wide by 24 pixels high on a transparent background:
   - Make the target dimensions fit within 22 pixels by 22 pixels and surround it with 1 pixel of padding.

3. Copy the row of icons and paste the duplicates underneath your original row in the same artboard so that your icons are in a consistent grid.
4. Create a style for the icon when the feature is in the default state:
   a. Apply the gray color #878a8c to an object on the artboard.
   b. Select that object.
   c. Create a graphic style and name it iconBaseColor.
5. Create a style for the icon when the feature is in the active, selected state:
   a. Apply the blue color #008abf to an object on the artboard.
   b. Select that object.
   c. Create a graphic style and name it iconSelectedColor.
6. Ungroup all of the objects on your artboard so that you can apply the styles that you created. Graphic styles have no effect on grouped objects.
7. Select all of the default state icons and apply the iconBaseColor style. For example, select the top row of icons to be the default versions of the icons.
8. Select all of the selected state icons and apply the iconSelectedColor style. For example, select your bottom row of icons to be the selected versions of the icons.
9. Save the file as an AI file so that you can edit this file in the future.
10. Save a copy of the file to use in your application:
    a. Select SVG as the file type.
    b. Remove the word copy from the end of the file name.
    c. In the SVG Options dialog box, ensure that you specify the following options:

      **Tip:** The following settings apply to Adobe Illustrator. If you are using a different vector-based drawing program, your SVG options might have different names.

<table>
<thead>
<tr>
<th>SVG options</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SVG</td>
</tr>
</tbody>
</table>
11. Verify the SVG code:
   a. Click **SVG Code**. Verify that the style tag has the following content:
      ```html
      <style type="text/css">
        .iconBaseColor{fill:#878A8C;}
        .iconSelectedColor{fill:#008ABF;}
      </style>
      ```
      Confirm that there are exactly two classes in the style tag.
   b. If there is any additional code in the style tag, verify that all objects are ungrouped and that all objects are applied with one of the graphic styles, and then check the SVG code again.

12. Include the SVG file in the `pluginPackage/WebContent` subdirectory of the JAR file for the plug-in that contains the feature for which you created the icon.

13. Implement the SVG code in your plug-in by continuing the steps for creating a feature. See “Creating a feature” on page 32.

**What to do next**

If you want to create more feature icons, edit the AI file and apply the `iconBaseColor` style and `iconSelectedColor` style for the icons that you add, and save your AI file. Save your changes as an SVG file and overwrite the old SVG file.

**Parent topic:** "Creating the plug-in components" on page 28

---

### Creating a layout

You can create a plug-in to provide a custom layout to use in IBM Content Navigator. The layout defines the arrangement of the widgets that are used in a desktop.

**Procedure**

To create a layout:

1. Extend the `ecm.widget.layout.MainLayout.js` class to create the primary widget for the layout:
   a. Implement the `getAvailableFeatures` method to return the features that are available in the layout. Each feature is declared as an instance of the `ecm.model.Feature` class.
   b. Implement the `setFeatures` method to configure how the features are arranged in the layout.

2. Extend the `PluginLayout.java` class. Implement the `getLayoutClass` method to return an instance of the widget class that you created in step 1.

3. Add the layout as a component in your `Plugin.java` subclass. To add the feature, implement the `getLayouts` method to return an instance of the `PluginLayout.java` class that you created in step 2.
Sample plug-in files for a custom layout

The following files in the sample plug-in define a layout:

Table 10. Sample files that define a plug-in layout

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampleLayout.js</td>
<td>This file extends the MainLayout.js class to create the widget that is used for the sample plug-in layout.</td>
</tr>
<tr>
<td>SamplePluginLayout.java</td>
<td>This file extends the PluginLayout.java class to define sample plug-in layout.</td>
</tr>
</tbody>
</table>

What to do next

After you add the plug-in in the web client, use the IBM Content Navigator administration tool to associate the layout with a desktop.

Parent topic: “Creating the plug-in components” on page 28

Related information:

- Class ecm.widget.layout.MainLayout
  Provides the main layout for IBM Content Navigator or for similar desktop layouts. This class displays the launch bar that contains the features on the left side of the layout.

- Class PluginLayout
  Provides an abstract class that is extended to define a layout that is provided by a plug-in. The layout is the main widget for the desktop and defines the overall arrangement of widgets in the desktop. The particular layout that is used for a desktop is selected in the IBM Content Navigator administration tool.

- Class Plugin
  Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.

Creating a repository type

You can create a plug-in for a custom repository type to authenticate, search, and provide result sets for repository servers. For example, you might want to use a custom repository type to extend IBM Content Navigator for your client application to use repositories that do not store documents. The custom repository type enables repository-specific authentication and other actions for your repository server.

Before you begin

Decide whether to use the custom repository type feature or use CMIS to support the connection between IBM Content Navigator and your custom repository.

- The custom repository type feature is a proprietary interface that uses the IBM Content Navigator JSON API. It was initially intended to support communication between internal Enterprise Content Management products and IBM Content Navigator.
- CMIS is an open standards interface. It is recommended that you use an open standards interface rather than a proprietary interface. If you decide to use CMIS, more help is available in the OpenCMIS Server Development Guide.
About this task

If you decide to use the custom repository type proprietary interface instead of using CMIS, follow the steps that are provided in this task.

Procedure

To define a repository type:

1. Create an internal identifier for the repository. The identifier is used for the return value for the PluginRepositoryType.getId method and the PluginRepositoryConnection.getType method.
2. Create a name for the repository. The name is displayed in the user interface.
3. Extend the PluginRepositoryType class to describe the repository type:
   a. Implement the getId method to return the internal identifier that you created in step 1.
   b. Implement the getName method to return the name that you created in step 2.
   c. Implement the login method to handle any repository-specific logic for desktop login. The login action is separate from all other actions so that it can be started from the desktop login when the repository is the authenticating repository.
   d. Implement the performAction to handle an action for the repository. The action is passed as a request parameter. The return parameter is a JSON response that is built by the logic in this method. You can use com.ibm.ecm.json classes to make this task easier.
4. Extend the PluginRepositoryConnection class to describe the repository connection:
   a. Implement the getName method to return the internal identifier that you created in step 1.
   b. Implement the getUserId method to return the user identifier that was entered at login.
   c. Implement the getUsername method to return the long name of the user for the banner.
5. Create subclasses that extend the classes in the Java script files PluginRepositoryGeneralConfigurationPane.js and PluginRepositoryConfigurationParametersPane.js to create a general configuration pane, and a configuration parameters pane for configuring your custom repository.

Sample plug-in files for a custom repository type

The following files in the sample plug-in define a repository type for a repository server that does not store documents.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginRepositoryType.java</td>
<td>This file extends the PluginRepositoryType.java class to create a repository type for a repository server that does not store documents.</td>
</tr>
</tbody>
</table>
What to do next

After you add the plug-in, you can select the repository type name from the menu for Create repository in the IBM Content Navigator administration tool. When you configure your repository, select the custom repository type for the plug-in and enter values in the general configuration pane and the configuration parameters pane that you created in step 5 on page 37.

Parent topic: “Creating the plug-in components” on page 28

Related reference:

IBM Content Navigator JSON API reference
The JSON API includes the requests and responses that can be used for a custom repository type.

Related information:

OpenCMIS Server Development Guide in PDF format
Downloads information about implementing CMIS on top of your custom data source.

Class PluginRepositoryConnection
Provides an abstract class that is extended to define a repository connection provided by the plug-in.

Class PluginRepositoryType
Provides an abstract class that is extended to define a repository type provided by the plug-in.

Class Plugin
Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.

Creating a request or response filter

You can create a plug-in to filter a request that is made to a service or to filter a response that is received from a service. For example, you might create a request filter to modify the parameters that are sent to a service to provide custom validations and error handling. You can also create a request filter to override a service. For example, you might create a response filter to apply custom formatting to the data that is received from a service before the data is displayed.

About this task

The external data service plug-in uses request and response filters to enable a service to provide custom choice lists and validation.

To use a filter, a plug-in must handle the JSONArtifact object that is sent in the request or the JSONObject object that is returned in the response.

Important: The JSON format that is used for request and response filters is subject to change without notice. If you create request and response filters for plug-ins, you might need to make updates to your code if a new version of the IBM Content Navigator Java application programming interface is released.

Procedure

To create a request or response filter:

1. If you are creating a request filter, extend the PluginRequestFilter class:
a. Implement the `getFilteredServices` method to return the names of the services for which this filter is used.

b. Implement the `filter` method to define the filter that is to be applied to the request.

**Tip:** Examine the JSON object that is sent to the service to determine the requirements for the filter method.

2. If you are creating a response filter, extend the `PluginResponseFilter` class:

a. Implement the `getFilteredServices` method to return the names of the services for which this filter is used.

b. Implement the `filter` method to define the filter that is to be applied to the request.

**Tip:** Examine the JSON object that is returned by the service to determine the requirements for the filter method.

3. Add the filter as a component in your `Plugin.java` subclass:

<table>
<thead>
<tr>
<th>Type of filter</th>
<th>Add to the <code>Plugin.java</code> subclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Implement the <code>getRequestFilters</code> method to return an instance of the <code>PluginRequestFilter.java</code> class that you created in step 1 on page 38.</td>
</tr>
<tr>
<td>Response</td>
<td>Implement the <code>getResponseFilters</code> method to return an instance of the <code>PluginResponseFilter.java</code> class that you created in step 2.</td>
</tr>
</tbody>
</table>

**Sample plug-in files for a custom response filter**

The following files in the sample plug-in define a response filter.

**Table 12. Sample files that define a plug-in response filter**

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginResponseFilter.java</td>
<td>This file extends the <code>PluginResponseFilter</code> class to define a filter that applies custom property formatting to search results.</td>
</tr>
</tbody>
</table>

**Parent topic:** "Creating the plug-in components" on page 28

**Related information:**

- **Class `PluginRequestFilter`**
  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.

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

You can create a plug-in to define a service that performs operations on a content server. For example, you can create a service that provides a viewer for specific document types on an IBM FileNet P8 server.

About this task

You use the application programming interfaces (APIs) that are native to the content server to define the operations that are performed by the service.

Tip: Follow best practices for servlets when you implement an IBM Content Navigator plug-in service:

- Because the application server is multithreaded, the plug-in service needs to handle multiple threads safely.
- The service will be called on multiple threads for multiple users; therefore, do not use instance variables. Using instance variables can have side effects such as unintentional sharing of information among users.

Procedure

To create a service:

1. Extend the PluginService class to create the service:
   a. Implement the execute method to provide the service with access to the APIs for the content server.
   b. If this service replaces one of the IBM Content Navigator services, implement the getOverriddenService method to identify that service.
   c. If the service requires configuration, use the PluginServiceCallbacks.loadConfiguration method to obtain the values that were configured for the service.

2. If the service provides a custom viewer, extend the PluginViewerDef class to define the URL template to launch the viewer.

3. Add the service as a component in your Plugin.java subclass:
   a. Implement the getServices method to return an instance of the PluginService.java class that you created in step 1.
   b. If the service provides a custom viewer, implement the getViews method to return an instance of the PluginViewerDef class that you created in step 2.

Sample plug-in files for a custom service

The following files in the sample plug-in define a service:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginService.java</td>
<td>This file extends the PluginService class to define a service that obtains system-related details about a document. To obtain this information, the service invokes the appropriate API depending on whether the document is in an IBM Content Manager, IBM Content Manager OnDemand, or IBM FileNet P8 repository.</td>
</tr>
</tbody>
</table>
Table 13. Sample files that define a plug-in service (continued)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginViewerDef.java</td>
<td>This file extends the PluginViewerDef class to define the URL template to launch the viewer that is defined in the SamplePluginViewerService.java file.</td>
</tr>
<tr>
<td>SamplePluginViewerService.java</td>
<td>This file extends the PluginService class to define a service that provides a custom viewer that does simple formatting of an archived RSS feed.</td>
</tr>
</tbody>
</table>

What to do next

Call the service as needed from your custom actions, viewers, or features. You then use the IBM Content Navigator to associate these components with a desktop, which makes the service available to users.

Parent topic: “Creating the plug-in components” on page 28

Related concepts:

- **IBM Content Navigator midtier services** on page 17
  The IBM Content Navigator services provide the connections to the repositories. In addition, the services support features such as search and document viewing across the repositories.

Related information:

- **Class PluginService**
  Provides an abstract class that is extended to create a class implementing each service provided by the plug-in. Services are actions, similar to servlets or Struts actions, that perform operations on the IBM Content Navigator server. A service can access content server application programming interfaces (APIs) and J2EE APIs.

- **Class PluginViewerDef**
  Provides an abstract class that is extended to define a viewer provided by the plug-in. The viewer can be used in viewer mappings to identify the types of documents for which the viewer should be invoked. Viewers are launched within their own iframes or web browser windows, based on a URL that is built by IBM Content Navigator by using a template that is defined in this class.

- **Class Plugin**
  Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.

Creating a viewer

You can create a plug-in to provide a custom viewer. Administrators can then map the custom viewer to document types by using the IBM Content Navigator administration tool.

Before you begin

Read and understand the steps to creating plug-ins to IBM Content Navigator.
About this task

IBM Content Navigator comes with a default viewer map to open and manipulate documents. You can also configure the AFP Viewer plug-in or configure the IBM Content Collector plug-in to view certain types of documents.

To provide a custom viewer of your own, create a plug-in to specify the custom viewer.

Procedure

To create a custom viewer:
1. Extend the PluginService class to create a service that provides the viewer.
2. Extend the PluginViewerDef class to provide a URL template that will open the viewer.
3. Extend the DocViewer class to provide a custom document viewer.
4. Optional: Customize the title that is displayed in the toolbar in this widget so that it blends in with your business scenario. Extend the ContentView class and use the getContentViewerTitle method and contentViewerTitle to assign your custom label.

Sample plug-in files for a custom viewer

The following files in the sample plug-in define a viewer:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePluginViewerDef.java</td>
<td>A definition of a new custom viewer. This class defines the sample viewer service as a viewer. The launch URL pattern describes how to invoke the viewer and provides the parameters that the viewer service uses to retrieve and format the document for viewing.</td>
</tr>
<tr>
<td>SamplePluginViewerService.java</td>
<td>This class extends the PluginService.java class to implement a service. The service provides a custom viewer that performs simple formatting of an archived RSS feed.</td>
</tr>
<tr>
<td>SampleImageViewer.java</td>
<td>This class extends the DocViewer.js class to implement a custom image viewer.</td>
</tr>
</tbody>
</table>

What to do next

After you write the plug-in, you must register and configure the plug-in in the IBM Content Navigator administration tool. To use the custom viewer, you must add the registered plug-in to a viewer map.

Parent topic: “Creating the plug-in components” on page 28

Related information:

- **Class Plugin**
  Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.
Class PluginService
Provides an abstract class that is extended to create a class implementing each service provided by the plug-in. Services are actions, similar to servlets or Struts actions, that perform operations on the IBM Content Navigator server. A service can access content server application programming interfaces (APIs) and J2EE APIs.

Class PluginViewerDef
Provides an abstract class that is extended to define a viewer provided by the plug-in. The viewer can be used in viewer mappings to identify the types of documents for which the viewer should be invoked. Viewers are launched within their own iframes or web browser windows, based on a URL that is built by IBM Content Navigator by using a template that is defined in this class.

[IBM Redbooks: Customizing and Extending IBM Content Navigator]
Find more information about customizing and extending IBM Content Navigator. For more information about integrating another viewer into IBM Content Navigator, see “Integrating another viewer into IBM Content Navigator.”

Creating a widget
You can create a widget to provide the user interface for your plug-in. For example, you might create a widget that enables users to run a custom action. You also might create a widget to enable administrators to configure the plug-in.

About this task
You can reuse the widgets in the IBM Content Navigator visual widget library in the user interface for your custom actions, features, layouts, and services. For example, rather than creating your own tree widget to browse a folder hierarchy, you can use the ecm.widget.FolderTree widget.

If no IBM Content Navigator widget meets your requirements, you can create a custom widget. The classes in the IBM Content Navigator JavaScript modeling library that make it easier to build new widgets from the basic Dojo dijit widgets. This library includes models of Dojo data stores that provide common methods for representing data. These data store model classes enable multiple widgets to easily use and present data. The library also includes models of Dojo trees that you can use to obtain data for navigation trees in custom widgets.

Procedure
To create a custom widget:
1. If your widget is extending the Dojo dijit._Templated class, create an HTML template to define the visual presentation of the widget, including buttons, fields, and labels. This template is used to generate the initial DOM structure for the widget.
2. Create a JavaScript file to define the class for the custom widget by extending the Dojo dijit._widget class or another Dojo dijit class.
   Use the Dojo dijit._Templated class and reference the template that you created in step 1.
3. Create a JavaScript file to instantiate the widget in the plug-in.
4. Add the widget script as a component in your Plugin.java subclass. To add the script, implement the getScript method to return the JavaScript file that you created in step 3.
Sample plug-in files for custom widgets

The following files in the sample plug-in define custom widgets:

Table 15. Sample files that define custom widgets

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessagesDialog.js</td>
<td>This file extends the Dojo dijit.Dialog class to define widget that provides a message dialog box.</td>
</tr>
<tr>
<td>MessagesDialog.html</td>
<td>This file provides an HTML template that is used for the widget that is defined in the MessagesDialog.js file.</td>
</tr>
<tr>
<td>PopupDialog.js</td>
<td>This file extends the Dojo dijit.Dialog class to define a widget that provides a pop-up dialog box.</td>
</tr>
<tr>
<td>PopupDialog.html</td>
<td>This file provides an HTML template that is used for the widget that is defined in the PopupDialog.js file.</td>
</tr>
<tr>
<td>SamplePlugin.js</td>
<td>This file instantiates the MessagesDialog.js class to display progress messages for the sample plug-in. The file instantiates the PopupDialog.js class to open the help file for the sample plug-in in a pop-up window.</td>
</tr>
</tbody>
</table>

What to do next

Use the widget to provide a user interface for the custom actions or features that are defined by your plug-in.

Parent topic: “Creating the plug-in components” on page 28

Related information:

- **Class Plugin**
  Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.

- **Writing Your Own Widget**
  See this web page for detailed information about creating and modifying widgets by using the Dojo Toolkit.

Packaging the plug-in components

You extend the Plugin class to identify the components that are used for a custom plug-in. You then create a JAR file to package the components deploy to IBM Content Navigator.

Procedure

To package the plug-in:

1. Create a JAR file that contains the files that are used for the plug-in.
   
   The JAR file must use the following structure:
   ```java
   packageName
   // Include all the Java classes that extend the Plugin class and
   // any other Java classes that are required by the plug-in
   Java class 1
   ```
In the packageName folder, include the Java class files that are defined for the plug-in. Create the subfolders based on the package names of the Java classes. Follow the standard JAR file structure by using nested folders for nested packages.

**Important:** All the Java classes must be included in a named package. You cannot use the default package.

2. Modify the `manifest.mf` file for your plug-in to include the following property:

```
Plugin-Class: pluginClassName
```

`pluginClassName` is the name of your `Plugin.java` subclass.

You can also include any of the standard properties for a JAR file in the `manifest.mf` file.

Parent topic: “Creating plug-ins to IBM Content Navigator” on page 25

Related information:

- [Class Plugin](#)

Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, as well as actions, services, and scripts provided by the plug-in.
External data services

You can use the external data service (EDS) representational state transfer (REST) protocol to create an external data service to get data from an external source, such as a file or a table in a database, to customize field properties and manage property behavior in IBM Content Navigator and IBM Content Navigator for Microsoft Office. Review the data flow diagram to understand how an external data service submits and returns requests between the IBM Content Navigator client and the external data source.

When you create an external data service, your existing data is integrated with IBM Content Navigator field values and other field properties, but you continue to store and maintain the data only in the original, authoritative data source. You access the data without moving or copying that data to a separate repository, so the source remains in the original data store. The external data source must remain available to IBM Content Navigator, so that the external data can be accessed whenever the business user invokes the service through the web client.

An advantage to using an external data service is that you do not need to modify the IBM Content Navigator source code to customize the user interface properties and values. Therefore, upgrades and other major changes to IBM Content Navigator do not affect the property data that is obtained by an external data service, because the data and external data service are located separately from IBM Content Navigator source code.

Where external data services can be implemented

An external data service can be implemented for the following actions in IBM Content Navigator or IBM Content Navigator for Microsoft Office:

- Adding documents and folders
- Checking documents in to the repository
- Editing properties for a selection of multiple items
- Editing item properties in the viewer
- Using entry templates
- Setting process workflow step properties and workflow filter criteria fields (IBM FileNet P8)
- Creating or using searches
- Controlling Content Manager OnDemand search folders

In these areas, you can use an external data service to customize the following field properties and property behaviors:

Look up values in a database to create choice lists
Create choice lists by using existing data that is managed in a different content repository or data source than the one that is connected to IBM Content Navigator.

For example, you can use values in a file that is located and managed in an external server or repository.

Prefill properties
Specify prefilled properties and default values.
For example, you can prefill fields with custom default values that are based on a particular class ID, authenticated user, or the parent folder.

**Specify property dependencies**
Define dependencies between properties.
For example, you might specify a dependency between a geographic region choice list property and an office branch choice list property so that when a user chooses a geographic region, the subsequent choice list that is dependent on the selected geographic region contains only the office branches that pertain to that geographic region.

**Set minimum and maximum values**
Specify an integer, float, or date to define the maximum or minimum value for a property.

**Restriction:** You cannot reset the minimum value or maximum value to be less restrictive than the minimum value or maximum value that is specified in the repository that you are using.

For example, if the minimum value in the repository is 100, the service can set the value to 150, but not to 50.

**Set read-only status**
Set a property to be read-only.
For example, you might create a property that requires a particular value. To prevent users from entering a different value that could cause an error, you can specify the correct default value and make that property read-only.

**Set required status**
Set a property to be a required field. When you use this attribute on a property, an asterisk appears in the user interface to indicate that the field is required. Users cannot proceed from the page or dialog box unless the field contains a value.

**Set hidden status**
Hide a property from the user interface.
For example, you might create a choice list that dynamically determines subsequent text input fields to present in a form. To hide a property that does not apply in a particular situation, you can use the hidden attribute.

**Implement property validation and error checking**
Show a custom message or provide assistance when users enter values into a property field.

**Restriction:** You cannot use custom validation for object properties, reference attributes, read-only properties, hidden properties, or search criteria.

**External data service architecture and data flow**

When an external data service is implemented for a certain action or property, the service is invoked when a business user interacts with that item in the web client. The following diagram shows how an external data service submits and returns requests.
The external data service works by using two services, which the application developer needs to create:

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

The external data service is invoked when a user opens a certain dialog box or tries to set a value on a property in the user interface. The middle tier services sends a request payload to the external data service, which interrogates the external data source for the requested information about the classes and attributes. The information from the response payload is then merged with the underlying information in the repository that you are using, for example, IBM Content Manager or FileNet P8.

**Precedence order in which data appears in the user interface**

In IBM Content Navigator or IBM Content Navigator for Microsoft Office, the following list describes the precedence order:

- The EDS-specified value takes precedence.
- With no EDS-specified value, the entry template value takes precedence.
With no EDS-specified value, and no entry template value, the EDS initial value takes precedence.

With no EDS-specified value, no entry template value, and no EDS initial value, the class attribute default value is displayed.

The terms in the previous list are defined as:

**EDS-specified value**
A change to the property value that resulted from an explicit assignment to the property value through EDS.

**Entry template value**
The default value provided by an entry template.

**EDS initial value**
The default value to be used as a prefill value for a property, which is defined through EDS.

**Class attribute default value**
The default value that is assigned to the property in the repository class definition.

**Restriction:** For IBM Content Navigator for Microsoft Office with Microsoft Outlook, precedence order includes only the entry template value and the class attribute default value. Property values cannot be modified by EDS. For example, EDS cannot override the Microsoft Outlook Email Document Title value, which uses the subject line of the email.

"Creating an external data service for IBM Content Navigator"
Use the EDS REST protocol to create an external data service that specifies requests to get data from an external data source to customize field properties and manage property behavior in IBM Content Navigator. You implement an external data service as a web application.

"Search criteria behavior for EDS implementations of dependent choice lists" on page 53
You can implement dependent choice lists in search criteria in IBM Content Navigator by using an external data service (EDS) or IBM Content Manager foreign keys. However, dependent choice lists that are provided by an EDS or foreign keys in search criteria behave somewhat differently from how dependent choice lists behave elsewhere in IBM Content Navigator.

"External data service REST protocol specifications" on page 54
Understand the specifications for the object type request, the particular object type request, and the responses that you need to create for each request.

"Sample external data service” on page 70
In the sample external data service, the values for a choice list are used from an external data source.

**Creating an external data service for IBM Content Navigator**
Use the EDS REST protocol to create an external data service that specifies requests to get data from an external data source to customize field properties and manage property behavior in IBM Content Navigator. You implement an external data service as a web application.

**Before you begin**
Understand which properties and areas in IBM Content Navigator you can customize through an external data service.
Review the specifications for defining an object types request and a particular object type request.

You need the following skills to implement an external data service:
- Java
- Java 2 Platform, Enterprise Edition
- Implement a web application with two services, for example, two Java servlets
- Read and write JSON

**About this task**

As you design your service, consider the following tips to improve the usability of your choice lists and other field properties:

**Reduce the size of choice lists by adding properties that can narrow subsequent lists**
For example, you can use questions to narrow the scope of the list, or use dependencies between properties to break long choice lists into shorter choice lists.

**Create dependent choice lists**
Dependent choice list values are replaced with a different list depending on the value that is selected in the preceding parent list. For example, you can set up Region and Branch Office choice lists, where the Branch Office list depends on the Region list selection. If the user selects a different region, the Branch Office list is cleared and populated by a different list of branch offices.

**Implement custom validation so that users can immediately resolve any errors in data entry.**
For example, if you implement text fields that have character restrictions, instead of using a custom validation error message to display only a description of the error, you can also provide users with a list of the unsupported characters so that users can avoid typing those characters in the future.

**Procedure**

To create an external data service:

1. Create a service that gets the list of classes to be handled by the external data service. For workflow information, you create a service that gets the list of registered service types instead of classes.
   
   For example, you might create a GetObjectTypes service:
   
   - Request: GET /types?repositoryId=repository_Id.
   - Response:
     
     ```
     [{
       "symbolicName": "object_type_or_service_type"
     }
     // More object types
     ]
     ```

   This service is invoked with the GET request, which provides the repository ID. The response in this example is an array of name values in a JSON file. This list must contain all the classes that are to be supported by the external data service.
2. Create a service that gets current attributes and values for each class that is included in the service that you created in the previous step. This service gets current attributes and values for a class and then returns information about those properties to the middle tier services. The data that is posted is JSON, and the response is also JSON.

For example, you might create an UpdateObjectType servlet that handles a POST method with a URL of the form /type/object_type_name:

- Request:

  ```
  POST /type/class_name_or_item_type_name
  {
    "repositoryId": "target_repository",
    "objectId": "if_an_existing_instance,_the_GUID_or_PID",
    "requestMode": "indicates_context_that_info_is_being_requested",
    "externalDataIdentifier": "opaque_identifier_meaningful_to_service",
    "properties": [
      {
        "symbolicName": "symbolic_name",
        "value": "the_current_value",
      }
      // More properties ...
    ]
  }
  ```

- Response:

  ```
  {
    "externalDataIdentifier": "opaque_identifier_meaningful_to_the_service",
    "properties": [
      {
        "symbolicName": "symbolic_name",
        "choiceList": {
          "displayName": "display_name",
          "choices": [
            {
              "displayName": "name",
              "value": value
            },
            // More choices in the choice list
          ]
        },
        // More properties
      }
      // More properties
    ]
  }
  ```

**What to do next**

After you write the external data service, you must deploy and start your service in the web application server, and register the EDSSupportPlugin plug-in in the IBM Content Navigator administration tool.

**Parent topic:** [External data services](#)

**Related concepts:**

- [IBM Content Navigator midtier services](#)

The IBM Content Navigator services provide the connections to the repositories. In addition, the services support features such as search and document viewing across the repositories.

**Related tasks:**

- [Registering and configuring plug-ins](#)
Related reference:

"Example: GET method request and response“ on page 56
This sample code submits a GET request to an external data service. The service then returns the supported object types, which includes the classes, workflow information, or item types for whose properties are to be updated with the values that are obtained from the external data source.

"Example: POST method request and response” on page 69
This sample code submits a request to an external data service when an IBM Content Navigator user selects a value for the region property, Region. The service then updates the choice list for the branch office property, BranchOffice, which depends on the Region property.

Related information:

IBM Redbooks: Customizing and Extending IBM Content Navigator
Find more information about customizing and extending IBM Content Navigator.

---

Search criteria behavior for EDS implementations of dependent choice lists

You can implement dependent choice lists in search criteria in IBM Content Navigator by using an external data service (EDS) or IBM Content Manager foreign keys. However, dependent choice lists that are provided by an EDS or foreign keys in search criteria behave somewhat differently from how dependent choice lists behave elsewhere in IBM Content Navigator.

**Restriction:** EDS support and IBM Content Manager foreign key support for dependent choice lists in search criteria apply only to single-class searches. Dependent choice lists that are provided through EDS and foreign keys cannot be used in cross-repository searches, multiple class searches, or the IBM Content Manager All pseudo-class.

**Runtime behavior when users specify a parent property multiple times in the search criteria**

If a property that has dependent (child) choice lists is added to the search criteria more than once, the last parent property whose value is updated controls the child choice list. For example, if the Country property is specified twice in the search criteria, first with the value United States and then later in the list with the value Brazil, the dependent State choice list will contain the choice list for states in Brazil. Users should therefore specify parent properties of dependent choice lists only once in a search.

To enable users to specify more than one value for a parent property, you can implement EDS so that it can handle multiple value updates for properties that have dependent choice lists. Then, users can use smart operators such as Include Any to specify multiple values for a property that has dependent choice lists. For example, if you use the Include Any operator and add United States and Brazil to the Country property, then the State property receives a choice list containing the states in the United States and Brazil.
Runtime behavior when users update the parent value of a dependent choice list

When users update the value of a parent property that has a child choice list, the selected value in the child choice list either persists or is cleared depending on whether the current value is available as an option in the new choice list.

The following behaviors occur with the current value in the child choice list when users update the value of the parent property:

- If the current child choice list value is among the values in the child choice list, the current child choice list value remains selected when the parent property is modified.
- If the current child choice list value is not available among the values that are in the new child choice list, the child choice list value is cleared (no value is selected) when the parent property is modified.

Parent topic: "External data services" on page 47

External data service REST protocol specifications

Understand the specifications for the object type request, the particular object type request, and the responses that you need to create for each request.

Object types resource

The object type resource represents the classes, item types, workflow step processors, or workflow in-basket filters to get for the external data service.

GET method

The GET request gets the list of object types for the external data service.

Parent topic: "External data services" on page 47

GET method

The GET request gets the list of object types for the external data service.
URI syntax

For the response, you create the object types resource, for example, an `ObjectTypes.json` file, which contains the class names, item type names, or workflow information for whose properties are to be updated with the values that are obtained from the external data source.

For IBM FileNet P8 workflow information, multi-part String values are used instead of a class or item type name.

```
/types?repositoryId=Repository_ID
```

The repository ID is the symbolic name of the data source that contains the class names, item type names, or workflow information that is used in the service.

Request content

```
GET /types?repositoryId=Repository_ID
```

Response content

The response to the request must include a JSON payload that contains the following parameters:

```json
{
  "types":
  [
    {
      "symbolicName": "TypeOne",
    },
    //More object types
    ...
  ]
}
```

Response codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 OK</td>
<td>The method completed successfully. The response that is returned by the GET method includes the JSON payload that contains the list of object types.</td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>One of the required parameters was missing or a parameter value was invalid.</td>
</tr>
<tr>
<td>404 Not Found</td>
<td>Class names or item type names were not found in the <code>repositoryId</code> data source. This error indicates that the external data service does not manage any property values for the class because the object types request is not implemented by this external data service. The EDS REST protocol does not return an error to IBM Content Navigator. IBM Content Navigator ignores the external data service and does not call the service for the particular object type for any class.</td>
</tr>
<tr>
<td>500 Internal Server Error</td>
<td>A server error occurred. For information about the error, see the <code>userMessage</code> element in the JSON response.</td>
</tr>
</tbody>
</table>

Parent topic: "Object types resource" on page 54

Related reference: "External data service integration into FileNet P8 workflows" on page 56

The external data service (EDS) can be implemented in data fields for process in-basket filter criteria, in step processors, and in launching workflows. You must construct a multi-part String that finds the service and identifies the in-basket filter, step, or workflow.
**External data service integration into FileNet P8 workflows**

The external data service (EDS) can be implemented in data fields for process in-basket filter criteria, in step processors, and in launching workflows. You must construct a multi-part String that finds the service and identifies the in-basket filter, step, or workflow.

You can implement an EDS in workflow data fields for the following areas:
- Process in-basket filter criteria
- Launch workflow and step processor properties

The EDS must register multiple-part values to identify the service request type. For workflow information, the multiple-part values are used instead of the class or item type name.

**Process in-basket filter criteria**

Construct the multi-part string to identify the in-basket filter that you are using in your EDS implementation. Use the following format to identify the in-basket filter:

*application_space_name.process_role_name.in-basket_name*

Example:

{"symbolicName": "DefaultApplication.Personal_Items.My_Personal_Work"}

**Launch workflow and step processor**

Construct the multi-part string to identify the launch workflow step or other steps in the step processor that you are using in your EDS implementation. Use the following format to identify the step:

*workflow_name.sheet_name.step_name*

Example:

{"symbolicName": "Data_Field_Workflow.Workflow.LaunchStep"}

The workflow registry supports an optional fourth part to the tokenized, multi-part string:

*workflow_name.sheet_name.step_name.EDS_Service_value*

The *EDS_Service_value* value comes from the value of an exposed custom data field that is called *EDS_Service* on that particular step. The workflow designer must manually create the *EDS_Service* data field and specify for that data field to be exposed for steps that require the *EDS_Service* value for EDS processing. When a workflow exposes this string on a step, the string is not displayed in the processors properties user interface; instead, the string is a hidden property.

For example, this string identifies a step called *Approval_Step* with the *EDS_Service* value of *MyService*:

{"symbolicName": "Data_Field_Workflow.Workflow.Approval_Step.MyService"}

**Parent topic:** [Object types resource](#)

**Example: GET method request and response**

This sample code submits a GET request to an external data service. The service then returns the supported object types, which includes the classes, workflow...
information, or item types for whose properties are to be updated with the values that are obtained from the external data source.

Request example
GET /types?repositoryId=Repository_ID

Response example
[
  {"SymbolicName": "Book"},
  {"SymbolicName": "Invoice"},
  {"SymbolicName": "Folder"},
  {"SymbolicName": "Document"}
]

Parent topic: "Object types resource” on page 54

Particular object type resource

The particular object type resource represents the properties for which property values are obtained from an external data source. When the user creates a new item or modifies an item of the particular object type, the EDS REST protocol uses the particular object type resource to obtain data for the corresponding class, item type, workflow step processors, or workflow in-basket filters from the external data source.

The EDS REST protocol calls the POST method for resources when an item is created or a property is modified. The service then returns the required information in the response to the POST request.

"POST method”
The POST method obtains data from an external data source for item types of a specific class. You do not call this method directly. Instead, the EDS REST protocol calls this method automatically when workflow information or a content item is being added or modified.

"Request modes” on page 61
When an item is created or modified in the web client, IBM Content Navigator calls the POST method for the particular object type resource to submit a request to the external data service. This request payload contains a request mode that indicates the action that is being performed.

"Response payload to a POST method request payload” on page 62
The external data service responds to a POST method that was submitted by the EDSSupportPlugin plug-in. The response payload contains values for the properties that are managed by the service.

"Error responses for an external data service” on page 69
If the POST method call fails, the response code that the external data service returns indicates the type of error that occurred. Write the error responses for your external data service.

"Example: POST method request and response” on page 69
This sample code submits a request to an external data service when an IBM Content Navigator user selects a value for the region property, Region. The service then updates the choice list for the branch office property, BranchOffice, which depends on the Region property.

Parent topic: "External data service REST protocol specifications” on page 54

POST method
The POST method obtains data from an external data source for item types of a specific class. You do not call this method directly. Instead, the EDS REST protocol calls this method automatically when workflow information or a content item is being added or modified.
When IBM Content Navigator calls the POST method, the request payload contains the current value for each property. The current value can be one of the following values:

- The default value
- The value persisted for the property in the repository
- The working value that the user entered for the property

The response payload that the external data service returns includes changes to the properties that it manages. The service can modify attributes of properties in addition to modifying property values.

IBM Content Navigator then merges these changes with the base data in the repository and returns the data to IBM Content Navigator.

**URI syntax**

```
/type/{class name}
```

### Path elements

**Table 17. Path elements for the POST method**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{object type name}</td>
<td>String</td>
<td>The symbolic name of the class name or item type name that defines the property that is being updated.</td>
</tr>
</tbody>
</table>

### Request content

```json
{
    "repositoryId":"target_object_store_name",
    "objectId":"GUID_or_PID_of_item_that_is_being_edited",
    "requestMode":"request_context",
    "externalDataIdentifier":"identifier_for_the_service",

    "properties":
    [
        {
            "symbolicName":"property_name",
            "value":current_value,
        }
    ],

    // More properties ...
    ],

    "clientContext":
    {
        "Key1":"Value1",
        "Key2":"Value2"
    }
}
```

**Table 18. Request parameters for the POST method**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repositoryId</td>
<td>String</td>
<td>Yes</td>
<td>The symbolic name of the target external data store that contains the property data.</td>
</tr>
<tr>
<td>objectId</td>
<td>String</td>
<td>No</td>
<td>The globally unique identifier (GUID) or persistent identifier (PID) that identifies the item that is being edited.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| requestMode    | String   | Yes       | One of the following request modes that indicates the reason that the POST method is being called:  
  - initialNewObject  
  - initialExistingObject  
  - inProgressChanges  
  - finalNewObject  
  - finalExistingObject  
  See [Request modes](#) for information about the request modes. |
| externalData   | String   | Yes       | A string that indicates the state of the data that was returned by the external data service. The request must include this identifier if the requestMode parameter is set to one of these values:  
  - inProgressChanges  
  - finalNewObject  
  - finalExistingObject |
| properties     | Array    | Yes       | An array that contains values for the properties that are defined for the class or item type. For each property, the request contains the symbolic name and the property value. |
Table 18. Request parameters for the POST method (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientContext</td>
<td>Array</td>
<td>No</td>
<td>An array that contains the following key-value pairs:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>userid</strong> The user ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>locale</strong> The locale.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>desktopid</strong> The desktop ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This parameter is used to send information to an external data service when</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>an IBM Content Navigator user begins to add a document, add a folder, use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>an entry template, or create a search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Restriction:</strong> The clientContext parameter does not include user</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>credentials. Your EDS implementation is invoked by IBM Content Navigator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>without propagating the user’s session and credentials. EDS is therefore</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>not suitable to implement in cases where you need user credentials to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>connect to your external system, for example, to filter results (choice</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>list values) by the logged in user. To do something with EDS that requires</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>access to the content server, you can implement a custom response filter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The clientContext array can also contain the following key-value pairs,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>which were added in IBM Content Navigator, Version 2.0.3 to support the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ability to include the active entry template’s ID and name in the EDS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>context so that EDS implementations can return a different response if an</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>entry template is active:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>objectStoreId (FileNet P8)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>String ID of the object store where the item is or will be stored</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>entryTemplateId (FileNet P8 and IBM Content Manager)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The active entry template IBM Content Navigator id string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>entryTemplateName (FileNet P8 and IBM Content Manager)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The active entry template name string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>entryTemplateVsId (FileNet P8)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The active entry template version series ID string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>entryTemplateItemId (IBM Content Manager)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The active entry template item ID string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If custom properties are set in the new</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ecm.model.HasAttributesMixin.retrieveDependentAttributeDefinitionsContext</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>object, those custom properties are also present in the clientContext</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>parameter.</td>
</tr>
</tbody>
</table>

Response codes

Table 19. Response codes for the POST method

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 OK</td>
<td>The method completed successfully. The response that is returned by the POST</td>
</tr>
<tr>
<td></td>
<td>method includes the updated information for the property.</td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>One of the required parameters was missing or a parameter value was invalid.</td>
</tr>
<tr>
<td>404 Not Found</td>
<td>The class that was specified in the request was not found. This error does</td>
</tr>
<tr>
<td></td>
<td>not indicate that the class is invalid. Instead, it indicates that the</td>
</tr>
<tr>
<td></td>
<td>external data service does not manage any property values for the class.</td>
</tr>
<tr>
<td></td>
<td>The EDS REST protocol does not return an error to IBM Content Navigator.</td>
</tr>
<tr>
<td>500 Internal Server Error</td>
<td>A server error occurred. For information about the error, see the userMessage element in the JSON response.</td>
</tr>
</tbody>
</table>
When an item is created or modified in the web client, IBM Content Navigator calls the POST method for the particular object type resource to submit a request to the external data service. This request payload contains a request mode that indicates the action that is being performed.

**Request modes**

When an item is created or modified in the web client, IBM Content Navigator calls the POST method for the particular object type resource to submit a request to the external data service. This request payload contains a request mode that indicates the action that is being performed.

You must configure the external data service to respond with the data that is required for that action. For example, if the request is to add a document, when the Add Document wizard is opened, the service needs to respond with the initial property values that are defined for the Add Document class.

The `requestMode` parameter indicates the action that is being performed in IBM Content Navigator. This action determines the response that is returned by the external data service.

For input payloads, the following values are provided on the context ID:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>The user ID for the user who is logged in to IBM Content Navigator.</td>
</tr>
<tr>
<td>locale</td>
<td>The locale of the browser that initiated this call to the external data service.</td>
</tr>
<tr>
<td>desktop</td>
<td>The desktop ID for the desktop that is in use.</td>
</tr>
</tbody>
</table>

The `requestMode` parameter can have the following values:

**initialNewObject**

Indicates that a new object is being created. For each property, the input payload contains the symbolic name and the default value that is defined in the repository.

The `initialNewObject` request mode is called only for the first object that is added of a particular class. The EDS information that is returned from the `initialNewObject` request is used when adding subsequent documents of the same class.

**initialExistingObject**

Indicates that an existing object is being edited. For each property, the input payload that is passed to the service contains the symbolic name and the value that is currently stored in the repository.

The input payload also contains the `objectId` parameter that specifies the globally unique identifier (GUID) or persistent identifier (PID) of the content item that is being edited. The service can use the GUID to refer to the content item. However, remember that the values stored in the
repository for the class can change. Therefore, the values that are provided in the input payload might not match the values that are currently stored in the repository for the class.

The input payload does not contain the `externalDataIdentifier` parameter. Instead, this parameter is set by the external data service and returned in the response payload. Subsequent requests made during the update of the class include the `externalDataIdentifier` parameter to indicate the current state of the data to the service.

**inProgressChanges**
Indicates that the external data service is being called in response to changes in one or more properties that have dependent properties.

This request mode is called before any changes take effect when the user changes property values. This call happens only in cases where dependent properties are being used by the EDS.

The input payload can contain the following information:

- The current working value for each property in the class or item type
- The `externalDataIdentifier` parameter, which indicates to the service the previous state of any properties that it updated
- For an existing class, the `objectId` parameter, which specifies the GUID or the PID of the content item that is being edited

The external data service responds to this request if the attributes or working value of any property that it manages changed. The service also responds to return a custom validation error.

**finalNewObject**
Before the object is persisted in the repository, this value indicates that the external data service is being called for the final time in the sequence of exchanges to create the object. After this call, the new object is created and the property values are persisted in the repository.

For each property, the input payload that is passed to the service contains the working values for all properties that are defined by the class.

**finalExistingObject**
Indicates that the external data service is being called for the final time in the sequence of exchanges to update an existing property. After this call, the updated property values are persisted in the repository.

For each property, the input payload that is passed to the service contains the working values for all properties that are defined by the class or item type.

**Parent topic:** [Particular object type resource” on page 57](#)

**Response payload to a POST method request payload**
The external data service responds to a `POST` method that was submitted by the `EDSSupportPlugin` plug-in. The response payload contains values for the properties that are managed by the service.

**Response content**
The response to the request must include a JSON payload that contains the following parameters:

```json
{
   "externalDataIdentifier": "opaque_identifier_meaningful_to_the_service",
```
"properties": [
{
  "symbolicName": "symbolic_name",
  "value": "potential_new_value",
  "customValidationMessage": "Description of an invalid reason",
  "customInvalidItems": [0,3,4,8], // invalid multi-value items
  "displayMode": "readonly_or_readwrite",
  "required": true_or_false,
  "hidden": true_or_false,
  "maxValue": overridden_maximum_value,
  "minValue": overridden_minimum_value,
  "maxLength": underlying_maximum_length,

  "choiceList": {
    "displayName": "display_name",
    "choices": [
      {
        "displayName": "name",
        "value": value
      },
      {
        "displayName": "name",
        "value": value
      },
      // More choices ...
    ]
  }

  "hasDependentProperties": true_or_false,
}

// More properties ...
]
Table 21. Response parameters for the POST method

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>externalDataIdentifier</td>
<td>String</td>
<td>No</td>
<td>The identifier provides contextual information to indicate the state of the data that the service is returning.</td>
</tr>
<tr>
<td>properties</td>
<td>Array</td>
<td>Yes</td>
<td>An array that contains values for the properties that are managed by the external data service. For each property, you can specify the symbolic name and the attributes, such as value, choice list, and maximum length.</td>
</tr>
</tbody>
</table>

**Property attributes**

The Properties parameter contains the following attributes for each property that is managed by the external data service. The external data service can determine many of these values dynamically so that the service can return a different value in each response.

Table 22. Attributes of properties in the response payload

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbolicName</td>
<td>String</td>
<td>Yes</td>
<td>The symbolic name of the property. The name must match the symbolic name that was specified in the request payload.</td>
</tr>
<tr>
<td>value</td>
<td>Determined by setting in the class</td>
<td>No</td>
<td>The value of the property. The value that is set by the external data service must correspond to the data type that is specified for the property in the class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The external data service can dynamically determine the property value based on the values of another other property.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the service does not specify a value, the current working value for the property is unchanged.</td>
</tr>
</tbody>
</table>
Table 22. Attributes of properties in the response payload (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom Validation</td>
<td>String</td>
<td>No</td>
<td>A message that describes why a property value is invalid.</td>
</tr>
<tr>
<td>Error</td>
<td></td>
<td></td>
<td>You can configure the external data service to validate the current value of a property. If the value is invalid, the service can leave the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value unchanged and return an error message in the customValidation parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example, the service might determine that an account number is invalid. However, you do not want the service to replace the account number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Instead, you can configure the service to return an error message in the customValidationError parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If this parameter is included in the response, the property value is deemed invalid. However, the absence of this attribute indicates only that</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the parameter passed the validation that is performed by the external data service. The value might still be invalid based on attributes that are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>not validated by the service. Values are validated by using regular expressions. For example, the following regular expression can be used to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>validate a simple email address:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;\b[\w.-]+@[\w.-]+.\w{2,4}\b&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Restriction:</strong> You can use custom validation for properties in workflow properties, teamspaces, and entry template properties. You cannot</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>use custom validation for object properties, reference attributes, read-only properties, hidden properties, or search criteria.</td>
</tr>
<tr>
<td>custom Invalid Items</td>
<td>Array of indexes</td>
<td>No</td>
<td>An array of indexes for a list of values for a multi-valued property.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When the external data service validates a multi-valued property, it can return this parameter to indicate the specific values that are invalid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If a multi-valued property is invalid and this parameter is not set, the property value as a whole is considered invalid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This attribute is applicable only if the customValidationError parameter indicates that the property is invalid.</td>
</tr>
<tr>
<td>displayMode</td>
<td>String</td>
<td>No</td>
<td>A string that specifies whether IBM Content Navigator is to display the property value as read-only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The external data service can set this parameter to one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>readonly</strong>                                                                         The user can view the property value but cannot modify it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>readwrite</strong>                                                                         The user can modify the property value. This setting is the default value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the property value is set to readonly in the class, the external data service cannot make the value writable. In this situation, a value of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>readwrite is ignored.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>format</td>
<td>String</td>
<td>No</td>
<td>An expression that describes the correct format for values to enter into the property. Uses <code>formatDescription</code> to provide a description for the format parameter.</td>
</tr>
<tr>
<td>format</td>
<td>String</td>
<td>No</td>
<td>The description that is displayed in a tooltip if the user enters a format that does not match the expression specified in the format parameter.</td>
</tr>
<tr>
<td>required</td>
<td>Boolean</td>
<td>No</td>
<td>A Boolean value that is set to true to indicate that a value is required for the property. The external data service can dynamically determine this setting based on the values of other properties. If this parameter is not specified, the value specified in the class is used. <strong>Restriction:</strong> You can override a hidden property to be shown by using <code>hidden:false</code>. However, IBM CMIS providers filter properties that are defined as hidden, so the property definition is not available to the CMIS connector in IBM Content Navigator. Therefore, you cannot use EDS to override properties that are defined as hidden in a CMIS-provided repository.</td>
</tr>
<tr>
<td>hidden</td>
<td>Boolean</td>
<td>No</td>
<td>A Boolean value that is set to true to indicate that the property is to be hidden in IBM Content Navigator. The external data service can dynamically determine this setting based on the values of other properties. If this parameter is not specified, the value specified in the class is used.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Integer, float, or date-time</td>
<td>No</td>
<td>A number that indicates the maximum value of the property. The external data service can dynamically determine this setting based on the values of other properties. If a maximum value is specified for the property in the repository, the service cannot make the setting less restrictive. That is, the service can set the maximum only to a smaller value. It cannot increase the maximum value. For example, if the maximum value in the repository is 100, the service can set the value to 50, but not to 150.</td>
</tr>
<tr>
<td>minValue</td>
<td>Integer, float, or date-time</td>
<td>No</td>
<td>A number that indicates the minimum value of the property. The external data service can dynamically determine this setting based on the values of other properties. If a minimum value is specified for the property in the repository, the service cannot make the setting less restrictive. That is, the service can set the minimum only to a larger value. It cannot decrease the minimum value. For example, if the minimum value in the repository is 100, the service can set the value to 150, but not to 50.</td>
</tr>
</tbody>
</table>
Table 22. Attributes of properties in the response payload (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxLength</td>
<td>Integer</td>
<td>No</td>
<td>A number that indicates the maximum length of characters in the property value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The external data service can dynamically determine this setting based on the values of other properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If a maximum length is specified for the property in Content Engine, the service cannot make the setting less restrictive. That is, the service can set the maximum length only to a smaller value. It cannot increase the maximum length. For example, if the maximum length in Content Engine is 100, the service can set the value to 50, but not to 150.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>choiceList</td>
<td>Object</td>
<td>No</td>
<td>An array that defines a list of choices for the property value. The external data service can determine the choices in the list dynamically based on the values of other properties. The <code>choiceList</code> value can contain a flat list of choices:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;choiceList&quot;:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;displayName&quot;: &quot;display_name_for_the_choice_list&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;choices&quot;:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;displayName&quot;: &quot;display_name_for_a_specific_choice&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;value&quot;: value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>// More choices ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Important: Do not implement choice lists in the Content Engine definitions for the same properties that are associated with the choice lists that you provide by using the external data service. The <code>choiceList</code> value can also use the following special values to empty the choice list or to use the choice list that is defined in the class definition:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;choiceList&quot;: null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Removes the currently assigned choice list, so that no choice list is associated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If valid values are defined for the property, when there is no choice list, IBM Content Navigator dynamically creates a choice list from the valid values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>default</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;choiceList&quot;: &quot;default&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Uses the original choice list that is defined for the property in the class definition, if one exists. If the class definition does not define a choice list, then the null handling applies (handles the property as if there is no choice list).</td>
</tr>
<tr>
<td>hasDependentProperties</td>
<td>Boolean</td>
<td>No</td>
<td>A Boolean value that is set to true if other properties depend on the value of this property. When this parameter is set to true, the POST method is called to update the dependent properties based on the new value whenever this property is updated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this parameter is set to false.</td>
</tr>
</tbody>
</table>
Error responses for an external data service

If the POST method call fails, the response code that the external data service returns indicates the type of error that occurred. Write the error responses for your external data service.

For example, the response code 404 Not Found indicates that the method did not find a resource, such as the class or item type. The response code 400 Bad Request indicates that a required parameter was not provided or that an incorrect value was specified for a parameter.

The JSON response that is returned by the method contains additional information about the error condition. The following example shows the format that the response uses to provide that information:

```
#Response
HTTP/1.1 404 Not Found
Content-Type: application/json;charset-UTF-8
{
    "userMessage":
    {
        "text":"The specified object type is not a valid object type.",
    }
    "underlyingDetails":
    {
        "causes":
        {
            "more detailed message 1",
            "more detailed message 2"
        }
    }
}
```

Example: POST method request and response

This sample code submits a request to an external data service when an IBM Content Navigator user selects a value for the region property, Region. The service then updates the choice list for the branch office property, BranchOffice, which depends on the Region property.

Request example

```
POST /type/class_name_or_item_type_name

{
    "repositoryId":"ObjectStore1",
    "objectId":"{EC4D244B-5980-4...6978-8796-A5B4C3D2E1F0}",
    "requestMode": "inProgressChanges",
    "properties":
    [
        {
            "symbolicName": "Region",
            "value": "Western",
        }
    ],
    "clientContext":
    {
        "userid": "user1",
        "locale": "en_US",
        "desktop": "default"
    }
}
```
Response example
{
   "externalDataIdentifier": "opaque_identifier_meaningful_to_the_service",
   "properties": [
      {
         "symbolicName": "Region",
         "hasDependentProperties": true
      },
      {
         "symbolicName": "BranchOffice",
         "choiceList": {
            "displayName": "Western Branch Offices",
            "choices": [
               {
                  "displayName": "Los Angeles",
                  "value": "Los Angeles"
               },
               {
                  "displayName": "San Francisco",
                  "value": "San Francisco"
               },
               {
                  "displayName": "Salt Lake City",
                  "value": "Salt Lake City"
               },
               {
                  "displayName": "Seattle",
                  "value": "Seattle"
               }
            ]
         }
      }
   ]
}

Parent topic: “Particular object type resource” on page 57

Sample external data service

In the sample external data service, the values for a choice list are used from an external data source.

A sample external data service and JSON files, which are the sample data source, are available in the `ECMClient_installdir\samples\sampleEDSService\WEB-INF\` classes directory.

Files in the sample external data service

You can use the sample service as an example and as a starting point for your own external data service. The sample service consists of a `GetObjectTypes` servlet, an `UpdateObjectTypes` servlet, the object types resource JSON file, JSON files as the data source, and the `web.xml` deployment file.

Deploying the sample external data service” on page 72

You modify the sample external data service to be appropriate for your content classes and then deploy the service in the web application server.

Parent topic: “External data services” on page 47

Related reference:
“Samples for IBM Content Navigator” on page 19

The IBM Content Navigator software package includes several samples: a plug-in, the external data service plug-in, and a set of web pages. You can use these samples to create custom applications.

Files in the sample external data service

You can use the sample service as an example and as a starting point for your own external data service. The sample service consists of a `GetObjectTypes` servlet, an

IBM Content Navigator: Developing Applications with IBM Content Navigator APIs

70
UpdateObjectTypes servlet, the object types resource JSON file, JSON files as the
data source, and the web.xml deployment file.

The sample service consists of the following files:

GetObjectTypesServlet.java servlet
This servlet gets the list of all classes to handle in the external data service.

UpdateObjectTypeServlet.java servlet
For each class, this servlet gets the current attributes or values and returns
that information in the response payload.

ObjectTypes.json file
This JSON file determines which classes you are using for the external data
service.

JSON files as the data source
These JSON files are the external data sources for the sample EDS, and can
be used to understand the required format of the JSON. The
ECMClient_installdir\samples\sampleEDSService\WEB-INF\classes
directory contains the following sample JSON data source files:
  • Article
  • Book
  • Document
  • Email
  • Folder
  • Invoice
  • NOINDEX
  • DefaultApplication*
  • Data_Field_*

Important: These files are provided primarily to help you understand the
expected JSON format and to help you start creating the JSON for your
own external data service. To use the sample files for EDS deployment,
extensive configuration is required.

The JSON needs to match the classes, item types, and the corresponding
custom properties on your system. To use the provided JSON files in the
sample, the JSON files need to be modified so that the correct custom
properties are aligned with the actual classes and custom properties in
your IBM FileNet P8 or IBM Content Manager repository.
Similarly, to use the workflow JSON files (IBM FileNet P8), the JSON must
be modified to match your IBM Content Navigator workflows, including
the appropriate steps, step names, workflow data fields, and other custom
properties.

To apply the sample EDS on a workflow filter, the Application Space,
Role, Inbasket, and filter criteria must be set up accordingly.

A sample localized file is included in the sample to help illustrate that the
EDS can send locale information in the request. The EDS developer can
specify for the EDS to look at the locale information in the request and
return the response JSON in the proper correct translation.

web.xml file
The web.xml file provides configuration and deployment information for
the web application. This file must reside in the WEB-INF directory under
the context of the hierarchy of directories that exist for the external data
service.
Deploying the sample external data service

You modify the sample external data service to be appropriate for your content classes and then deploy the service in the web application server.

About this task

An Eclipse project is provided with the sample EDS service. You must modify the libraries that are referenced in the project to fit your configuration. For example, the \navigator\build\lib\j2ee\j2ee.jar might not be the correct path to your j2ee.jar file. Find the equivalent JAR file for your environment and update the project configuration to point to that JAR file.

Procedure

To deploy the sample external data service:

1. Copy the sample external data service from the installation directory into a dynamic web project in a development application such as Rational® Application Developer.
2. Modify the JSON files in the sample so that the correct custom properties are aligned with the actual classes and custom properties in your IBM FileNet P8 or IBM Content Manager repository.
3. Compile the project and build a WAR file. You can use the build.xml file that is included in the sample as a starting point.
4. Deploy the WAR file into your web application server.
5. Register and configure the external data service support plug-in, which is called edsPlugin, to use the sample external data service:
   a. Load the edsPlugin.jar file. In the IBM Content Navigator administration tool, in the configuration options for the plug-in, define the URL or the file path to the location of the edsPlugin.jar file that the IBM Content Navigator web application server can access:
      • Enter the path to the file that is installed on the local server, for example:
        - Microsoft Windows: C:\Program Files\IBM\ECMClient\plugins\edsPlugin.jar
        - Linux or AIX®: /opt/IBM/ECMClient/plugins/edsPlugin.jar
      The ECMClient folder is the IBM Content Navigator installation directory.
      • Enter the URL to the file that is in the deployed IBM Content Navigator path, for example: http://host_name:port_number/ICN_web_app_deployment_context/edsPlugin.jar
        To make the edsPlugin.jar file available in the deployed IBM Content Navigator path, you must copy the edsPlugin.jar file from the installation path (for example, C:\Program Files\IBM\ICN_installed_path\plugins) to the deployment path (for example, ICN_deployed_path\navigator.ear\navigator.war).
   b. After the edsPlugin.jar file is loaded, you must specify the URL to where the EDS service is deployed. For example, you can specify: http://host_name:port_number/sampleEDSService.
   c. Click Save and Close.
   d. Refresh your web browser.
Results

When you select values from choice list properties, the list contains the values from your external data source, which is specified by the external data service.

What to do next

In the IBM Content Navigator client, verify that the fields and properties are showing the correct data and are behaving as you expect.

Parent topic: [Sample external data service](#) on page 70
IBM Content Navigator API reference

IBM Content Navigator provides JavaScript and Java application programming interfaces (APIs) so that you can create custom applications. The JavaScript API includes the classes for the visual widget library and the modeling library. The Java API includes the classes that you can use to create plug-ins to IBM Content Navigator.

For screen reader users on Mozilla Firefox web browsers, traverse through the landmark navigation area in the JavaScript API reference documentation by using the Tab key.

"IBM Content Navigator modeling library structure"
The classes in the modeling library provide the business logic and data for IBM Content Navigator. These classes are used by the widgets to access and represent data in the content servers and in the IBM Content Navigator configuration.

"IBM Content Navigator visual widget catalog” on page 78
The JavaScript classes in the IBM Content Navigator visual widget library are grouped by package. In some packages, the classes can be further grouped by the type of widgets that they represent.

IBM Content Navigator modeling library structure

The classes in the modeling library provide the business logic and data for IBM Content Navigator. These classes are used by the widgets to access and represent data in the content servers and in the IBM Content Navigator configuration.

The following diagram shows the hierarchy of the primary classes in the modeling library:
As shown in this diagram, an instance of the Desktop class encompasses the other objects in the model. The Desktop object specifies the repositories, features, actions, and viewers to which a set of users will have access.

An instance of the Repository class represents a specific repository. The repository can be an IBM Content Manager or IBM Content Manager OnDemand server, or an IBM FileNet Content Engine object store.

The Repository object gives users the ability to access and perform actions on repository objects. These objects are also represented by classes in the modeling library:

SearchTemplate class
This class represents a search that is stored in the repository. In an IBM Content Manager or IBM FileNet P8 repository, a SearchTemplate object represents a saved search. In an IBM Content Manager OnDemand repository, a SearchTemplate object represents a folder.

A SearchTemplate object gives users the ability enter or modify the criteria that is used to perform a search. The criteria is represented in the model library by the SearchCriterion class.
UnifiedSearchTemplate class
This class represents a unified search template that can be stored in a repository and be used to search in multiple repositories simultaneously. To improve performance, the search criteria are not loaded initially. Instead, the retrieveSearchCriteria method must be called to initiate an asynchronous request to make them available.

EntryTemplate class
This class represents an entry template that is stored in an IBM FileNet P8 repository. An EntryTemplate object gives users the ability to create a document, folder, or custom object. An EntryTemplate object also provides the default values for the destination folder, properties, and security.

ContentClass class
This class represents a document or folder class in a IBM FileNet P8 repository or an item type in an IBM Content Manager repository. A ContentClass object gives users the ability to access the item and to edit the properties and attributes of the item.

Each property or attribute is represented by an instance of the AttributeDefinition class that contains information about the property or attribute, such as type and allowed values.

WorklistFolder class
This abstract class represents a collection of work lists, which are sometimes called in-baskets or inboxes. The subclasses of the WorklistFolder class represent collections of the different types of work lists and in-baskets. For example, the ProcessApplicationSpace class represents a collection of process roles that determine who can access an IBM FileNet P8 process application. The ProcessRole class represents a role that is defined on an IBM FileNet P8 server. An instance of this class determines who has access to the in-baskets that are defined in an application space.

Worklist class
This class represents a single work list. A Worklist object provides users with the ability to process the work items that are assigned to them.

The model includes the ProcessInBasket subclass for IBM FileNet P8 repositories.

Teamspace class
This class represents a teamspace. A Teamspace object provides users with the ability to organize and share the content that a team needs to complete its tasks.

The individual users and groups who belong to a teamspace are represented in the model by the User and UserGroup classes.

TeamspaceTemplate class
This class represents a teamspace template. A TeamspaceTemplate object provides users with the ability a set of predefined options for creating a teamspace.

ResultSet class
This class represents a set of items that are returned by a search. A ResultSet object provides users with the ability to locate and select documents, folders, or work items.

An individual item in a ResultSet object is represented in the model by the Item class or one of its subclasses:
• The `ContentItem` class represents a document, folder, or other content item in the repository.
• The `WorkItem` class represents a workflow item.

Other classes in the modeling library support the classes that are shown in the diagram. For example, the model includes the following classes:
• The `SearchTemplateFolder` class and `TeamspaceFolder` class represent collections of search templates and teamspaces. These abstract classes provide collections for items such as such the recent folders or all folders that are displayed in navigation trees in the user interface.
• For an IBM Content Manager OnDemand repository, a `SearchTemplateFolder` object represents a cabinet.
• The `Request` class represents a request that is made to an IBM Content Navigator service.
• The `PropertyFormatter` class represents the formatting that is applied to properties when they are displayed. This class can be extended or overridden to provide custom formatting of certain types of properties.
• The `_ModelStore` class and the classes that are suffixed with `TreeModel` represent Dojo data stores and trees. These classes can be used with Dojo dijit classes to populate widgets with data.

**Parent topic:** [IBM Content Navigator API reference](#) on page 75

**Related concepts:**
[IBM Content Navigator development architecture](#) on page 15
IBM Content Navigator uses a model-view-controller (MVC) architecture.

---

**IBM Content Navigator visual widget catalog**

The JavaScript classes in the IBM Content Navigator visual widget library are grouped by package. In some packages, the classes can be further grouped by the type of widgets that they represent.

**Widgets package** on page 79
The classes in the `ecm.widget` package define widgets that you use to create and edit repository objects such as documents, folders, and work items. This package also includes classes that define common user interface components such as buttons, menus, and tabs.

**Administration widgets package** on page 88
The classes in the `ecm.widget.admin` package define the widgets that make up the IBM Content Navigator administration tool. You can use these widgets to customize the administration user interface.

**Administration model package** on page 89
The classes in the `ecm.widget.admin.model` package provide the model for the user interface of the IBM Content Navigator administration tool. You can use the widgets in this package to extend the capabilities of the IBM Content Navigator administration tool. Alternatively, you can use these widgets to provide a separate administration user interface that is based on the IBM Content Navigator user interface.

**Content list widgets package** on page 89
The classes in the `ecm.widget.listView` package define widgets, modules, and decorators that are used to construct the content list seen in IBM Content Navigator. The content list is used to provide lists of repository items, favorites, work items, and so on.
The `ecm.widget.dialog` package contains classes that define the dialog boxes that are used in the IBM Content Navigator web client.

The `ecm.widget.layout` package contains classes that define components of the web client layout.

The `ecm.widget.process` package contains classes that are used to build custom launch processors and step processors for an IBM FileNet P8 workflow.

The classes in the `ecm.widget.search` package define widgets that you use to create and edit repository objects such as documents, folders, and work items.

The `ecm.widget.teamspaceBuilder` package contains the widgets that create and edit teamspaces and teamspace templates. In IBM Content Navigator, these widgets are used for the teamspace builder.

The `ecm.widget.viewer` package contains classes that are used to view document content.

**Widgets package**

The classes in the `ecm.widget` package define widgets that you use to create and edit repository objects such as documents, folders, and work items. This package also includes classes that define common user interface components such as buttons, menus, and tabs.

**Document and folder widgets**

These widgets are used to view and edit documents and folders.

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddContentItemGeneralPane</td>
<td>Provides a pane that is used in the AddContentItemDialog widget to add a document or folder to a repository. This pane is also used in the CheckInDialog widget to check a document in to a repository.</td>
</tr>
<tr>
<td>AddContentItemPropertiesPane</td>
<td>Provides a pane that is used in the AddContentItemDialog widget to edit properties when a user adds a document or folder to the repository. This pane is also used in the CheckInDialog widget to edit properties when a user checks a document in to the repository.</td>
</tr>
<tr>
<td>Widget class</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AddContentItemSecurityPane</td>
<td>Provides a pane that is used in the AddContentItemDialog widget to view and edit the security settings of a document or folder that is being added to a repository. This pane is also used in the CheckInDialog widget to view and edit the security settings of a document or folder that is being checked in to a repository.</td>
</tr>
<tr>
<td>ChildComponentPropertiesPane</td>
<td>Provides a widget that is used to view and edit the attributes of a child component of an IBM Content Manager item type.</td>
</tr>
<tr>
<td></td>
<td>A child item type can have unique attributes. For example, IBM Content Collector uses child components to store email instance information and references to attributes of the distinct email item type.</td>
</tr>
<tr>
<td>ChildComponentSelector</td>
<td>Provides a widget that is used to display the child components of an IBM Content Manager item type.</td>
</tr>
<tr>
<td>CommonPropertiesPane</td>
<td>Provides a widget that is used to view and edit properties. This widget can be used in dialog boxes for adding documents, creating folders, and editing properties.</td>
</tr>
<tr>
<td>ContentClassSelector</td>
<td>Provides a widget that contains a tree or drop-down list that is used to select the class for a document or folder.</td>
</tr>
<tr>
<td>ContentListDropDownDialog</td>
<td>Provides a widget that contains a drop-down list that is used in a content list.</td>
</tr>
<tr>
<td>ContentListEditPane</td>
<td>Provides a widget that supports an edit properties pane for a single document.</td>
</tr>
<tr>
<td>FolderSelector</td>
<td>Provides a widget that is used to select folders in a repository, teamspace, or parent folder.</td>
</tr>
<tr>
<td>FolderSelectorCallback</td>
<td>Provides callback functions for the folder selector widget.</td>
</tr>
<tr>
<td>FoldersFiledInPane</td>
<td>Provides a widget that shows the folders in which a document is filed.</td>
</tr>
<tr>
<td>FolderTree</td>
<td>Provides a widget that is used to browse the folders in a repository.</td>
</tr>
<tr>
<td>ItemCommentsPane</td>
<td>Provides a widget that is used to view comments of or add comments to an item. Also provides controls to edit and delete accessible comments.</td>
</tr>
<tr>
<td>ItemEditPane</td>
<td>Provides a widget that is used to edit a folder or a document in a repository.</td>
</tr>
<tr>
<td>ItemNotelogsPane</td>
<td>Provides a widget that is used to view notelogs of or add notelogs to an item. Also provides controls to edit and delete accessible notelogs.</td>
</tr>
</tbody>
</table>
Table 23. Widgets that are used to view and edit documents and folders (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemPreviewPane</td>
<td>Provides a widget that displays a preview of the content for an item.</td>
</tr>
<tr>
<td>ItemPropertiesDisplayPane</td>
<td>Provides a widget that displays a read-only view of the properties for an item.</td>
</tr>
<tr>
<td>ItemPropertiesPane</td>
<td>Provides a widget that is used to edit the properties of an item.</td>
</tr>
<tr>
<td>ItemPropertiesPaneExtension</td>
<td>Provides a widget that is used to extend the ItemPropertiesPane. The extension can add a new section, similar to the Properties and System Properties sections, which can contain custom properties for display and editing by the user.</td>
</tr>
<tr>
<td>ItemPropertiesStackPane</td>
<td>Provides a pane that contains a stack of ItemPropertiesPane widgets. Typically, the ItemPropertiesStackPane contains a single ItemPropertiesPane widget. However, for reference attributes, a separate ItemPropertiesPane widget is added to the stack for each reference attribute the user opens. The user sees the top widget in the stack.</td>
</tr>
<tr>
<td>ItemSecurityPane</td>
<td>Provides a widget that contains the SecurityPane widget and is used by the Edit Properties dialog. This class retrieves the item permissions and other item information that is required by the SecurityPane widget to display the security information.</td>
</tr>
<tr>
<td>SecurityPane</td>
<td>Provides a widget that contains a list of users and roles that have access to an item.</td>
</tr>
<tr>
<td>UnSelectableFolder</td>
<td>This class is used for folders that should not be selectable in the folder selector control.</td>
</tr>
<tr>
<td>VersionsPane</td>
<td>Provides a widget that is used to view the versions of an item.</td>
</tr>
</tbody>
</table>

Teamspace and user widgets

These widgets support the definition of teamspaces and the selection of users.

Table 24. Widgets that are used to define teamspaces and select users

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TeamList</td>
<td>Provides a widget that displays a list of the users and groups that are assigned to a teamspace.</td>
</tr>
<tr>
<td>UserGroupSelector</td>
<td>Provides a widget that is used to select a user or group. This widget can be configured to display both users and groups or only users. The widget can also be configured to support multiple selections or only single selections.</td>
</tr>
</tbody>
</table>
Table 24. Widgets that are used to define teamspaces and select users (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserGroupSelectorPane</td>
<td>Provides a pane that is used by the _PropertiesMixin class to select users or groups for assigning property value. The widget can be configured to display voting approval properties for Entry Template workflows.</td>
</tr>
</tbody>
</table>

User interface widgets

These widgets provide common user interface components.

Table 25. Widgets that are used to define common user interface components

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_DroppedFilesAddDocMixin</td>
<td>Provides methods that are used to handle content class attributes. ContentClass and SearchContentClasses mixes in _DroppedFilesAddDocMixin.</td>
</tr>
<tr>
<td>_HoverHelpMixin</td>
<td>Provides a base class for creating and displaying hover help in the user interface.</td>
</tr>
<tr>
<td>_MoveUpDownGridxMixin</td>
<td>Provides a base class for a widget that is used to move items up or down in a gridx widget.</td>
</tr>
<tr>
<td>_MoveUpDownMixin</td>
<td>Provides a base class for a widget that is used to move items up or down in a list.</td>
</tr>
<tr>
<td>_SecurityMixin</td>
<td>Provides a base class that contains methods for working with IBM FileNet P8 security data.</td>
</tr>
<tr>
<td>_Separator</td>
<td>Provides the separator that is used in the breadcrumb.</td>
</tr>
<tr>
<td>_SimpleBreadcrumb</td>
<td>Provides the data that is used in the breadcrumb.</td>
</tr>
<tr>
<td>_SinglePropertyEditorMixin</td>
<td>Provides a base class for a single property editor.</td>
</tr>
<tr>
<td>ActionMenu</td>
<td>Provides a widget that is used as a context menu.</td>
</tr>
<tr>
<td>Banner</td>
<td>Provides a banner that is displayed in the user interface for an IBM Content Navigator application.</td>
</tr>
<tr>
<td>BookmarkPane</td>
<td>Provides a widget that is provides a URL-addressable page that can display a bookmarked folder. The URL for the page can be created from the View Link action or the Email as Link action for a folder.</td>
</tr>
<tr>
<td>Breadcrumb</td>
<td>Provides a widget that displays the position of a user in the IBM Content Navigator application.</td>
</tr>
</tbody>
</table>
Table 25. Widgets that are used to define common user interface components (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Button</strong></td>
<td>Provides a widget that is used as a button in the user interface. This class extends the Dojo dijit.form.Button class to provide hover help for the button.</td>
</tr>
<tr>
<td><strong>CheckBox</strong></td>
<td>Provides a check box that can be used to select several values for the same property. This class extends the Dojo dijit.form.CheckBox class to provide hover help for the check box.</td>
</tr>
<tr>
<td><strong>ComboBox</strong></td>
<td>Provides a widget that displays a combination box that consists of a field and a drop-down list. The user can select a value from the list or type a value in the field. The value in the field does not need to match a value in the list. This class extends the Dojo dijit.form.ComboBox class to provide hover help for the combination box.</td>
</tr>
<tr>
<td><strong>CompositeButton</strong></td>
<td>This widget provides a button that can be configured to display a button icon on the left of the container, a button title in the middle, or an action button icon on the right. This button is used by the SecurityPane widget to display item security entries.</td>
</tr>
<tr>
<td><strong>DatePicker</strong></td>
<td>Provides a widget that is used to select a date. This class extends the idx.form.DateTimePicker class to provide additional formatting patterns.</td>
</tr>
<tr>
<td><strong>DesktopPane</strong></td>
<td>Provides a widget that is used to display a desktop.</td>
</tr>
<tr>
<td><strong>DownloadCount</strong></td>
<td>Provides a widget that is used to display the download count property for a document, including a control to retrieve and display the list of users that have downloaded the document.</td>
</tr>
<tr>
<td><strong>DropDownButton</strong></td>
<td>Extension to dijit.form.DropDownButton to add the idx.DropDownOpen CSS class whenever the DropDownList object is opened. This allows for alternate styling on the widget when its drop-down is in the open state. This is included with idx.ext.</td>
</tr>
<tr>
<td><strong>DropDownDialog</strong></td>
<td>Provides a widget that is used to display a drop-down list.</td>
</tr>
<tr>
<td><strong>DropDownInput</strong></td>
<td>Provides a widget that combines a drop-down list and a field. A user can enter a value in the field or select a value from the drop-down list.</td>
</tr>
<tr>
<td><strong>DropDownLink</strong></td>
<td>Provides a widget that contains a link with a drop-down menu and summary.</td>
</tr>
<tr>
<td>Widget class</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ellipsis</td>
<td>Provides a widget that displays an ellipsis (...) to indicate that data is truncated so the user is not seeing all the values. The widget can be configured to truncate data on the left or on the right. By default, data is truncated on the right.</td>
</tr>
<tr>
<td>FavoritesTree</td>
<td>Provides a widget that contains a tree that is used to display the user’s favorites.</td>
</tr>
<tr>
<td>FilteringSelect</td>
<td>Provides a combination box that consists of a drop-down list and a field. The user can select a value from the list or type a value in the field. The list is filtered to display values that begin with the value entered in the field. A valid entry must match completely a value from the list.</td>
</tr>
<tr>
<td></td>
<td>This class extends the Dojo dijit.form.FilteringSelect class to support hover help.</td>
</tr>
<tr>
<td>FilterTextBox</td>
<td>Provides a widget that is used to filter the input in a text box.</td>
</tr>
<tr>
<td>GlobalToolbar</td>
<td>Provides a widget that is used as a global toolbar for an IBM Content Navigator application.</td>
</tr>
<tr>
<td>HoverHelp</td>
<td>Provides a widget that is used to display the question mark icon that links to help in the user interface.</td>
</tr>
<tr>
<td>HoverHelpText</td>
<td>Provides a widget that is used to display hover help and text in a grid cell in the user interface.</td>
</tr>
<tr>
<td>LoginPane</td>
<td>Provides a widget that is used to log in to a content management server. This widget prompts the user for credentials to authenticate to the server.</td>
</tr>
<tr>
<td>MessageBar</td>
<td>Provides a widget that is used as a message bar in an IBM Content Navigator application.</td>
</tr>
<tr>
<td>MultiColumnList</td>
<td>Provides a simple multiple-column list that is used to select items.</td>
</tr>
<tr>
<td>MultiValueChoicePane</td>
<td>Provides a widget that is used to select multiple values from a choice list. This widget is used for IBM FileNet Content Manager repositories.</td>
</tr>
<tr>
<td>MultiValueInputPane</td>
<td>Provides a widget that is used to enter multiple values. This widget is used for IBM FileNet Content Manager repositories.</td>
</tr>
<tr>
<td>NumberSpinner</td>
<td>Provides a widget that contains a text box that is used to enter a numeric value. This class extends the Dojo dijit.form.NumberSpinner class to support hover help.</td>
</tr>
</tbody>
</table>
Table 25. Widgets that are used to define common user interface components (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NumberTextBox</strong></td>
<td>Provides a widget that contains a text box that is used to enter a numeric value. This class extends the Dojo dijit.form.NumberTextBox class to support hover help.</td>
</tr>
<tr>
<td><strong>PropertyEditors</strong></td>
<td>Provides a widget that holds form fields in a private array variable.</td>
</tr>
<tr>
<td><strong>RadioButton</strong></td>
<td>Provides a widget that contains a set of choices from which only one can be selected. This class extends the Dojo dijit.form.RadioButton class to support hover help.</td>
</tr>
<tr>
<td><strong>RangeBoundTextBox</strong></td>
<td>Provides a widget that contains a text box for which a range of valid values is specified. This class extends the Dojo dijit.form.RangeBoundTextBox class to support range checking for large exponents.</td>
</tr>
<tr>
<td><strong>ReferenceAttributeButton</strong></td>
<td>Provides a widget that is used as a button to display the object identified by a reference attribute in an IBM Content Manager repository.</td>
</tr>
<tr>
<td><strong>RepositorySelector</strong></td>
<td>Provides a widget that is used to select a repository.</td>
</tr>
<tr>
<td><strong>Select</strong></td>
<td>Provides a widget that is used as a button that displays a drop-down list when it is clicked. This class extends the Dojo dijit.form.DropDownButton class to provide hover help for the button.</td>
</tr>
<tr>
<td><strong>SelectObject</strong></td>
<td>Provides a widget that is used to select a repository object by entering a query.</td>
</tr>
<tr>
<td><strong>ShowHyperlinkPane</strong></td>
<td>Provides a widget that is used to display the hyperlink for a document or folder.</td>
</tr>
<tr>
<td><strong>SinglePropertyEditorFactory</strong></td>
<td>Provides a factory that creates a SinglePropertyEditor instance.</td>
</tr>
<tr>
<td><strong>SingleSelectTree</strong></td>
<td>Provides a widget that contains a tree in which the user can select one node in the tree at a time.</td>
</tr>
<tr>
<td><strong>SingleValueTreePane</strong></td>
<td>Provides a widget that contains a tree in which the user can select one value at a time. This widget is used for IBM FileNet Content Manager object stores.</td>
</tr>
</tbody>
</table>
### Table 25. Widgets that are used to define common user interface components (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SloshBucket</td>
<td>Provides a widget that is used to pick a subset of unique items from a larger set of items. The widget contains two single column lists. One list contains the available items and the other list contains the selected items. The <strong>Add</strong> button and a <strong>Remove</strong> button that are used to move items from one list to the other appear between the columns. Optionally, the widget includes an <strong>Up</strong> button and a <strong>Down</strong> button that are used to order the selected items.</td>
</tr>
<tr>
<td>StandByDropDownInput</td>
<td>Provides a widget that combines a drop-down list and a field. A user can enter a value in the field or select a value from the drop-down list. It also hooks up the stand by status when the drop-down pane is waiting for query data.</td>
</tr>
<tr>
<td>StateSelect</td>
<td>Provides a widget that contains a drop-down selection box. This class extends the Dojo <code>dijit.form.Select</code> class to support options that can be set programmatically for the selection box, but that are hidden from users.</td>
</tr>
<tr>
<td>TextBox</td>
<td>Provides a widget that contains a text box. This class extends the Dojo <code>dijit.form.TextBox</code> class to provide hover help for the text box.</td>
</tr>
</tbody>
</table>
Table 25. Widgets that are used to define common user interface components (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TreeSelector</td>
<td>Provides a widget that contains a tree that is used to navigate the folders in a repository. This widget allows users to select multiple folders, but it does not allow them to select nodes with no item value.</td>
</tr>
<tr>
<td>TreeSloshBucket</td>
<td>Provides a widget that contains a slosh bucket that uses a tree to display available and selected items.</td>
</tr>
<tr>
<td>ValidationTextarea</td>
<td>Provides a widget that contains a text area in which the user input is validated against the specified criterion. This class extends ecm.widget.ValidationTextBox and the Dojo dijit.form.SimpleTextarea class to add validation.</td>
</tr>
<tr>
<td>ValidationTextBox</td>
<td>Provides a widget that contains a text box in which the user input is validated against the specified criterion. This class extends the Dojo dijit.form.ValidationTextBox class to add validation for byte maximum length.</td>
</tr>
<tr>
<td>VerticalSloshBucket</td>
<td>Provides a vertical slosh bucket that is used in teamspace builder to add classes or entry templates, searches, and existing roles to teamspace templates.</td>
</tr>
</tbody>
</table>

Teamspace widgets

These widgets are used to view and edit teamspaces.

Table 26. Widgets that are used to view and edit teamspaces

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TeamspacePropertiesPanl</td>
<td>Provides a widget that is used to view or edit the properties of a teamspace template.</td>
</tr>
</tbody>
</table>

Workflow widgets

These widgets are used to view and edit workflows, work lists, and work items.

Table 27. Widgets that are used to view and edit workflows, work lists, and work items

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartWorkflow</td>
<td>Provides a widget that is used to start folders and documents on a workflow.</td>
</tr>
<tr>
<td>SuspendWorkItems</td>
<td>Provides a widget that is used to suspend folders and documents on a workflow.</td>
</tr>
<tr>
<td>WorklistSelector</td>
<td>Provides a widget that contains a tree that is used to navigate process workflow items. This widget displays the available work lists, process roles, and in-baskets for selection.</td>
</tr>
</tbody>
</table>

Parent topic: "IBM Content Navigator visual widget catalog” on page 78
Administration widgets package

The classes in the ecm.widget.admin package define the widgets that make up the IBM Content Navigator administration tool. You can use these widgets to customize the administration user interface.

General widgets

These widgets define the general layout of the administration tool.

*Table 28. Widgets that define the general layout of the administration tool*

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionMenu</td>
<td>Provides a context menu that is used in the IBM Content Navigator administration tool.</td>
</tr>
<tr>
<td>AdminGrid</td>
<td>Provides a scrollable, tabular grid that is used to display administration objects.</td>
</tr>
<tr>
<td>AdminLoginDialog</td>
<td>Provides a dialog box that is used to log in to a selected repository from the IBM Content Navigator administration tool.</td>
</tr>
<tr>
<td>AdminTabs</td>
<td>Provides a container for the IBM Content Navigator administration tool that has multiple panes with a tab that corresponds to each pane.</td>
</tr>
<tr>
<td>AdminTree</td>
<td>Provides a navigation tree for administration objects such as desktops, labels, menus, plug-ins, repositories, settings, and viewer maps.</td>
</tr>
<tr>
<td>NavigatorAdminPane</td>
<td>Provides the administration view for the application layout. This view is used for administration and configuration tasks.</td>
</tr>
</tbody>
</table>

Plug-in configuration widgets

These widgets support configuration of plug-ins.

*Table 29. Widgets that are used to configure plug-ins*

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PluginConfigurationPane</td>
<td>Provides a base class that can be extended to create a configuration interface for a plug-in.</td>
</tr>
</tbody>
</table>

Repository configuration widgets

These widgets are used to configure the repositories that IBM Content Navigator connects to.

*Table 30. Widgets that are used to configure repositories*

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODCustomPropertiesDialog</td>
<td>Provides a dialog box that is used to edit custom properties for the connection to an IBM Content Manager OnDemand repository.</td>
</tr>
</tbody>
</table>

Parent topic: “IBM Content Navigator visual widget catalog” on page 78
Administration model package

The classes in the `ecm.widget.admin.model` package provide the model for the user interface of the IBM Content Navigator administration tool. You can use the widgets in this package to extend the capabilities of the IBM Content Navigator administration tool. Alternatively, you can use these widgets to provide a separate administration user interface that is based on the IBM Content Navigator user interface.

These widgets define the model for the administration user interface.

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminTreeModel</td>
<td>Provides a widget that contains a hierarchical navigation tree for administration objects.</td>
</tr>
<tr>
<td>_AdminObjectBase</td>
<td>Provides the base class for all administrative objects.</td>
</tr>
</tbody>
</table>

Parent topic: [IBM Content Navigator visual widget catalog” on page 78](#)

Content list widgets package

The classes in the `ecm.widget.listView` package define widgets, modules, and decorators that are used to construct the content list seen in IBM Content Navigator. The content list is used to provide lists of repository items, favorites, work items, and so on.

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContentList</td>
<td>Provides a widget that displays the results of a search, the contents of a folder, or the work items in a work list. This widget can be used to navigate folders and launch actions for the items that are displayed.</td>
</tr>
<tr>
<td>Toolbar</td>
<td>Provides a widget that is used to provide toolbar capability to the content list.</td>
</tr>
<tr>
<td>Bar</td>
<td>Provides the bar capability, which allows users to arrange the content list widgets.</td>
</tr>
<tr>
<td>Breadcrumb</td>
<td>Provides the breadcrumb capability. This widget uses the <code>resultSet</code> class <code>parentFolder</code> method or <code>searchTemplate</code> method to get the start of the path. It then traverses the whole path using the parent on the item or search template. If the breadcrumb is clicked, it calls <code>contentList openItem</code>.</td>
</tr>
<tr>
<td>DocInfo</td>
<td>Provides the document information pane.</td>
</tr>
<tr>
<td>FilterData</td>
<td>Provides filter capability using the gridx module <code>Filter</code>. This module allows the user to type in their filter in the filter text box. This module will then filter the grid on any of the filterable columns containing this input value.</td>
</tr>
<tr>
<td>InlineMessage</td>
<td>Provides the ability to display error, warning, confirm, and information messages inline.</td>
</tr>
<tr>
<td>TotalCount</td>
<td>Provides the shown total count utility.</td>
</tr>
</tbody>
</table>
Table 32. Widgets in `ecm.widget.listView.modules` that are used to construct the content list (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewDetail</td>
<td>Provides the detail view capability.</td>
</tr>
<tr>
<td>ViewMagazine</td>
<td>Provides the magazine view capability.</td>
</tr>
<tr>
<td>ViewFilmStrip</td>
<td>Provides the filmstrip view capability.</td>
</tr>
</tbody>
</table>

The following grid modules are in the `ecm.widget.listView.gridModules` subdirectory:

Table 33. A subset of the grid modules in the content list

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowContextMenu</td>
<td>Provides a widget that displays the context menu when the user right-clicks an item or presses Shift+F10. This widget also provides default action capability.</td>
</tr>
<tr>
<td>DndFromDesktopAddDoc</td>
<td>Provides a widget that controls the drag-and-drop action from the desktop to the grid.</td>
</tr>
<tr>
<td>DndRowMoveCopy</td>
<td>Provides drag-and-drop capability for the row, checking for security when hovering over a row, and providing the move or copy action when dropping the document or documents.</td>
</tr>
<tr>
<td>DndRowCopy</td>
<td>Provides drag-and-drop capability for the row, checking for security when hovering over a row, and providing the copy action when dropping the document or documents.</td>
</tr>
<tr>
<td>DndDropOnly</td>
<td>Provides drop capability onto a row and checks for security when hovering over a row. This module extends ./DndRowMoveCopy. This widget adds the capability of disallowing the dragging of rows.</td>
</tr>
</tbody>
</table>

Parent topic: "IBM Content Navigator visual widget catalog" on page 78

**Dialog box widgets package**

The `ecm.widget.dialog` package contains classes that define the dialog boxes that are used in the IBM Content Navigator web client.

**Message dialog boxes**

These widgets define dialog boxes that are used to display messages.

Table 34. Widgets that are used to display messages

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorDialog</td>
<td>Provides a dialog box that is used to display an error message that was generated by a service request.</td>
</tr>
<tr>
<td>MessageDialog</td>
<td>Provides a dialog box that displays a simple message to the user.</td>
</tr>
</tbody>
</table>
Table 34. Widgets that are used to display messages (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusDialog</td>
<td>Provides a dialog box that is used to display a status message during the processing of a request to the server.</td>
</tr>
</tbody>
</table>

**Search dialog boxes**

These widgets define dialog boxes that support the searching.

Table 35. Widgets that are used to search for items

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SearchBuilderDialog</td>
<td>Provides a dialog box that is used to add search templates.</td>
</tr>
<tr>
<td>SearchDialog</td>
<td>Provides a dialog box that is used to open and run saved searches.</td>
</tr>
<tr>
<td>SearchSelectorDialog</td>
<td>Provides a dialog box that is used to select search templates.</td>
</tr>
</tbody>
</table>

**Selection dialog boxes**

These widgets define dialog boxes that are used to select objects and users.

Table 36. Widgets that are used to select items

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContentClassSelectorDialog</td>
<td>Provides a dialog box that is used to select content classes.</td>
</tr>
<tr>
<td>SelectObjectDialog</td>
<td>Provides a dialog box that is used to search for and select repository items.</td>
</tr>
<tr>
<td>SelectUserGroupDialog</td>
<td>Provides a dialog box that is used to select users and groups.</td>
</tr>
</tbody>
</table>

**Work item and workflow dialog boxes**

These widgets define dialog boxes that are used with work items and workflows.

Table 37. Widgets that are used with work items and workflows

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManageProcessRolesDialog</td>
<td>Provides a dialog box that is used to modify FileNet P8 process role members.</td>
</tr>
<tr>
<td>ReassignToUserDialog</td>
<td>Provides a dialog box that is used to reassign an FileNet P8 workflow item to another user.</td>
</tr>
<tr>
<td>StartWorkflowDialog</td>
<td>Provides a dialog box that is used to start documents or folders on a IBM Content Manager document routing workflow.</td>
</tr>
<tr>
<td>StepProcessorWindow</td>
<td>Provides a mechanism that is used to open the FileNet P8 step processor user interface in a separate browser window.</td>
</tr>
</tbody>
</table>
Table 37. Widgets that are used with work items and workflows (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StepRoutingWindow</td>
<td>Provides a mechanism that is used to display the IBM Content Manager Step Processor in a dialog.</td>
</tr>
<tr>
<td>SuspendWorkItemsDialog</td>
<td>Provides a dialog box that is used to suspend work items in a IBM Content Manager workflow.</td>
</tr>
<tr>
<td>TransferredWorkflowsDialog</td>
<td>Provides a dialog box that is used to launch a transferred workflow. Displays and enables user to select from a list of FileNet P8 workflows to launch.</td>
</tr>
<tr>
<td>WorkflowSubscriptionsDialog</td>
<td>Provides a dialog that is used to display FileNet P8 workflow subscriptions associated with the class of the target object to be used for selecting a FileNet P8 workflow to launch.</td>
</tr>
<tr>
<td>WorkflowPreferencesDialog</td>
<td>Provides a dialog box that is used to view FileNet P8 workflow user preferences. These values are saved to the Process Engine server and not the Admin database.</td>
</tr>
</tbody>
</table>

**Miscellaneous dialog boxes**

These widgets define various dialog boxes.

Table 38. Widgets that define dialog boxes

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AboutDialog</td>
<td>Provides a dialog box that displays information about the application such as version information.</td>
</tr>
<tr>
<td>AddAnnotationDialog</td>
<td>Provides a dialog box that is used to add annotations to a document in an IBM Content Manager OnDemand repository.</td>
</tr>
<tr>
<td>AddContentItemDialog</td>
<td>Provides a dialog box that is used to add documents or folders to a repository. Users can use this dialog box to specify all required parameters for adding documents or folders.</td>
</tr>
<tr>
<td>AddToFavoritesDialog</td>
<td>Provides a dialog box that is used to add a repository item as a favorite.</td>
</tr>
<tr>
<td>AnnotationDialog</td>
<td>Provides a dialog box that is used to add and view annotations for an item in an IBM Content Manager OnDemand repository.</td>
</tr>
<tr>
<td>ApplyRemoveHoldDialog</td>
<td>Provides a dialog box that is used to apply or remove holds on IBM Content Manager OnDemand documents.</td>
</tr>
<tr>
<td>BaseDialog</td>
<td>Provides the base dialog box from which all other ecm.widget.dialog dialog box classes are derived.</td>
</tr>
<tr>
<td>ChangePasswordDialog</td>
<td>Provides a dialog box that is used to change the password on an IBM Content Manager server or an IBM Content Manager OnDemand server.</td>
</tr>
</tbody>
</table>
Table 38. Widgets that define dialog boxes (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckInDialog</td>
<td>Provides a dialog box that is used to check documents in to a repository.</td>
</tr>
<tr>
<td>ConfirmationDialog</td>
<td>Provides a dialog box that displays a question and prompts the user to approve or cancel the action.</td>
</tr>
<tr>
<td>ContentViewerWindow</td>
<td>Provides a separate browser window that displays the ContentViewer widget.</td>
</tr>
<tr>
<td>CreateHoldDialog</td>
<td>Provides a dialog box that is used to create a hold on an IBM Content Manager OnDemand item.</td>
</tr>
<tr>
<td>EditPropertiesDialog</td>
<td>Provides a dialog box that is used to edit the properties of documents, folders, and content items.</td>
</tr>
<tr>
<td>LoginDialog</td>
<td>Provides a dialog box that is used to log in to a content server.</td>
</tr>
<tr>
<td>MoveFileDialog</td>
<td>Provides a dialog box that is used to move a document or folder from one folder to another folder.</td>
</tr>
<tr>
<td>PrintDialog</td>
<td>Provides a dialog box that is used to print one or more documents.</td>
</tr>
<tr>
<td>ShowHyperlinkDialog</td>
<td>Provides a dialog box that is used to display the hyperlink to a folder or the hyperlink to a specific version of a document.</td>
</tr>
<tr>
<td>UnfileDialog</td>
<td>Provides a dialog box that is used to remove a document or a folder from a folder.</td>
</tr>
<tr>
<td>YesNoCancelDialog</td>
<td>Provides a dialog box that displays a question and the buttons Yes, No, and Cancel.</td>
</tr>
</tbody>
</table>

Parent topic: “IBM Content Navigator visual widget catalog” on page 78

Layout widgets package

The ecm.widget.layout package contains classes that define components of the web client layout.

These widgets define components of the web client layout.

Table 39. Widgets that define components of the web client layout

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TabContainerBase</td>
<td>Provides the outermost tab container for a tabbed layout.</td>
</tr>
<tr>
<td>AdminPane</td>
<td>Provides a pane that is used to display the administration interface in the layout.</td>
</tr>
<tr>
<td>BaseLayout</td>
<td>Provides the base class from which all other layout widgets are derived.</td>
</tr>
<tr>
<td>BookmarkActionsHandler</td>
<td>Extends the CommonActionsHandler class to provide additional handlers for actions that are related to bookmarks.</td>
</tr>
</tbody>
</table>
### Table 39. Widgets that define components of the web client layout (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BookmarkLayout</td>
<td>Provides a desktop layout that is used to display bookmarks.</td>
</tr>
<tr>
<td>BrowseFlyoutPane</td>
<td>Provides a fly-out pane that is used to browse folders and documents.</td>
</tr>
<tr>
<td>BrowsePane</td>
<td>Provides a pane that is used to browse folders and documents.</td>
</tr>
<tr>
<td>CommonActionsHandler</td>
<td>Provides the default implementation for many of the common actions. This class invokes methods from the ecm.widget.dialog package and ecm.model package.</td>
</tr>
<tr>
<td>FavoritesPane</td>
<td>Provides a pane that is used to store and access the documents, folders, or predefined searches that a user uses most frequently.</td>
</tr>
<tr>
<td>FlyoutMenuContainerChild</td>
<td>Provides a container for the fly-out panes that are used for features.</td>
</tr>
<tr>
<td>HorizontalScrollPane</td>
<td>Provides a pane that can be scrolled horizontally.</td>
</tr>
<tr>
<td>LaunchBarContainer</td>
<td>Provides a vertical bar that displays the features available in a layout.</td>
</tr>
<tr>
<td>MainLayout</td>
<td>Provides the main layout for IBM Content Navigator or for similar desktop layouts. This class displays the launch bar that contains the features on the left side of the layout.</td>
</tr>
<tr>
<td>ManageTeamspacesPane</td>
<td>Provides a widget that displays a list of all teamspaces and teamspace templates that a user can access. From the list, a user with appropriate authority can open, edit, and delete teamspaces and teamspace templates. A user with the appropriate authority can also open the builder to create new templates and teamspaces from this widget.</td>
</tr>
<tr>
<td>NavigatorMainLayout</td>
<td>Extends the MainLayout class to provide additional interactions that are specific to IBM Content Navigator.</td>
</tr>
<tr>
<td>RecentActivityPane</td>
<td>Provides a pane that displays information about the recent activity on items.</td>
</tr>
<tr>
<td>SearchFlyoutPane</td>
<td>Provides a fly-out pane that contains the search interface for a layout.</td>
</tr>
<tr>
<td>SearchPane</td>
<td>Provides a pane that contains the search interface for a layout.</td>
</tr>
<tr>
<td>TeamspaceBuilderPane</td>
<td>Provides a pane to display the teamspace builder in the layout.</td>
</tr>
<tr>
<td>TeamspaceFlyoutPane</td>
<td>Provides a fly-out pane that is used for the teamspaces feature in the launch bar.</td>
</tr>
<tr>
<td>TeamspacePane</td>
<td>Provides a pane that is used to display a teamspace in the layout.</td>
</tr>
</tbody>
</table>
Table 39. Widgets that define components of the web client layout (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkFlyoutPane</td>
<td>Provides a fly-out pane that is used to select repositories and to navigate IBM Content Manager worklists and IBM FileNet P8 application spaces.</td>
</tr>
<tr>
<td>WorkPane</td>
<td>Provides a pane that contains workflow navigation components. This pane restricts repository selection to IBM FileNet P8 repositories with workflow privileges and IBM Content Manager repositories.</td>
</tr>
</tbody>
</table>

Parent topic: “IBM Content Navigator visual widget catalog” on page 78

Process widgets package

The ecm.widget.process package contains classes that are used to build custom launch processors and step processors for an IBM FileNet P8 workflow.

These widgets define launch processors and step processors and are used to view and edit workflows, work lists, and work items.

Table 40. Widgets that define launch processors and step processors

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_PatternPropertiesMixin</td>
<td>Provides a mixin that defines methods for working with pattern workflow properties pane.</td>
</tr>
<tr>
<td>_ProcessorMixin</td>
<td>Provides a mixin that defines methods for working with step processor metadata. This class creates the IBM FileNet P8 workflow response buttons. In addition, the class provides utility functions for working with filed data.</td>
</tr>
<tr>
<td>InbasketContainer</td>
<td>Provides a widget that is used to display the in-baskets that are associated with an IBM FileNet Process Engine role. Each in-basket is displayed in a ContentList widget that is contained in the InbasketTabContainer widget.</td>
</tr>
<tr>
<td>InbasketFilterContainer</td>
<td>Provides a widget that is used to display the query filters that are specified on the in-basket content. The filters can be used to refine the query for the items that are displayed in the in-basket.</td>
</tr>
<tr>
<td>InbasketFiltersPane</td>
<td>Provides a widget that is used to display the Filter and Reset buttons for working with the in-basket filters. In addition, this widget displays the number of filters that are currently in effect.</td>
</tr>
<tr>
<td>LaunchPatternLayout</td>
<td>Provides a layout that is used to launch pattern workflow processes.</td>
</tr>
<tr>
<td>LaunchPatternPropertiesPane</td>
<td>Provides a widget that is used to view or edit the properties of a pattern work item.</td>
</tr>
<tr>
<td>LaunchProcessorLayout</td>
<td>Provides a layout that is used to launch workflow processes.</td>
</tr>
</tbody>
</table>
Table 40. Widgets that define launch processors and step processors (continued)

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MilestonesLayout</td>
<td>Provides a layout that is used to display milestones for an IBM FileNet P8 workflow.</td>
</tr>
<tr>
<td>ProcessorActionButtonBar</td>
<td>Provides the Step processor custom action button bar for IBM Content Manager workflow processors.</td>
</tr>
<tr>
<td>ProcessorButtonBar</td>
<td>Provides the Step processor button bar for IBM FileNet P8 workflow processors.</td>
</tr>
<tr>
<td>ProcessorLayout</td>
<td>Provides the basic layout for IBM FileNet P8 workflow processors.</td>
</tr>
<tr>
<td>StepPatternLayout</td>
<td>Provides the basic layout for step processors.</td>
</tr>
<tr>
<td>StepPatternPropertiesPane</td>
<td>Provides a widget that is used to view or edit the properties of a pattern work item.</td>
</tr>
<tr>
<td>StepProcessorLayout</td>
<td>Provides the basic layout for step processors.</td>
</tr>
<tr>
<td>StepRoutingLayout</td>
<td>Provides a widget that is used to display the IBM Content Manager Step Processor.</td>
</tr>
<tr>
<td>TrackerHistoryPane</td>
<td>Provides a widget that is used to view the history of an IBM FileNet P8 workflow.</td>
</tr>
<tr>
<td>TrackerLayout</td>
<td>Provides the basic layout for the IBM FileNet P8 workflow tracker page. Users can use the tracker page to monitor the progress of a running workflow.</td>
</tr>
<tr>
<td>TrackerMilestonesPane</td>
<td>Provides a pane that is used to view the milestones for a tracker.</td>
</tr>
<tr>
<td>WorkItemAttachmentsPane</td>
<td>Provides a widget that is used to view or edit work item attachments.</td>
</tr>
<tr>
<td>WorkItemPropertiesPane</td>
<td>Provides a widget that is used to view or edit the properties of a work item.</td>
</tr>
</tbody>
</table>

Search widgets package

The classes in the ecm.widget.search package define widgets that you use to create and edit repository objects such as documents, folders, and work items.

These widgets provide the mechanisms for searching a repository.

Table 41. Widgets that are used to search repositories

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeDefinitionsForm</td>
<td>Provides a widget that is used by the BasicSearchDefinition widget to represent a collection of attributes within the scope of the selected class or classes that can be used to construct search criteria. Each attribute in the widget is an instance of the AttributeDefinitionWidget widget.</td>
</tr>
<tr>
<td>AttributeDefinitionWidget</td>
<td>Provides a widget that is used by the search builder to represent an attribute in a search template.</td>
</tr>
<tr>
<td>Widget class</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SearchBuilder</td>
<td>Provides a wrapper widget that contains the BasicSearchBuilder widget. This widget handles relationships with other widgets such as maintaining opened searches and closing the widget when a search is canceled. This widget also locks or unlocks a saved search when the search is opened or closed.</td>
</tr>
<tr>
<td>SearchBuilderDialog</td>
<td>Provides a dialog box that is used to add search templates.</td>
</tr>
<tr>
<td>SearchClassSelector</td>
<td>Provides a widget that is used by the search builder to provide the option to select multiple classes. Extends the ContentClassSelector widget.</td>
</tr>
<tr>
<td>SearchDialog</td>
<td>Provides a dialog box that is used to open and run saved searches.</td>
</tr>
<tr>
<td>SearchInCriteria</td>
<td>Provides a widget that displays the folder or repository to be searched by a saved search. For a saved search that was defined in IBM FileNet Workplace XT to support searching multiple folders or repositories, this widget includes a link to open a drop-down to see a list of the folders or repositories.</td>
</tr>
<tr>
<td>SearchInDropDown</td>
<td>Provides an interface to select the scope of a search.</td>
</tr>
<tr>
<td>SearchInDropDownInput</td>
<td>Provides a drop-down button for SearchInputDropDown widget.</td>
</tr>
<tr>
<td>SearchMoreOptions</td>
<td>Provides a widget that displays additional options for a search. The additional options include the object type, version, join relationship of the property, text conditions, file type, and action filters.</td>
</tr>
<tr>
<td>SearchPropertyOptions</td>
<td>Provides a widget that contains property search criteria in a drop-down dialog box. The dialog box can be defined to return results that match all conditions or any condition.</td>
</tr>
<tr>
<td>SearchResultsDisplayOptions</td>
<td>Provides a widget that contains the columns that are used to display search results. The search results can be sorted by a property in descending or ascending order. If the text search is enabled, the search results can be sorted by rank in descending order.</td>
</tr>
<tr>
<td>SearchSelector</td>
<td>Provides a widget that contains a tree or drop-down list that is used to select a saved search.</td>
</tr>
<tr>
<td>SearchSelectorDialog</td>
<td>Provides a dialog box that is used to select search templates.</td>
</tr>
<tr>
<td>SearchTab</td>
<td>Provides a widget that contains a SearchForm widget and a ContentList widget with the widget events wired together.</td>
</tr>
<tr>
<td>SearchTabContainer</td>
<td>Provides a widget that contains and controls the search tabs that are open.</td>
</tr>
</tbody>
</table>
Parent topic: IBM Content Navigator visual widget catalog on page 78

**Teamspace builder widgets package**

The `ecm.widget.teamspaceBuilder` package contains the widgets that create and edit teamspaces and teamspace templates. In IBM Content Navigator, these widgets are used for the teamspace builder.

These widgets are used to create a teamspace or teamspace template.

*Table 42. Widgets that are used to create teamspaces and teamspace templates*

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TeamspaceBuilder</td>
<td>Provides the widget that contains the user interface for teamspace builder.</td>
</tr>
</tbody>
</table>

Parent topic: IBM Content Navigator visual widget catalog on page 78

**Viewer widgets package**

The `ecm.widget.viewer` package contains classes that are used to view document content.

These widgets define dialog boxes that are used to view content.

*Table 43. Widgets that are used to view content*

<table>
<thead>
<tr>
<th>Widget class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AjaxViewer</td>
<td>Provides a widget that supports page-by-page viewing of documents.</td>
</tr>
<tr>
<td>AppletViewer</td>
<td>Provides the IBM Content Manager OnDemand Viewer Applet widget.</td>
</tr>
<tr>
<td>BrowserViewer</td>
<td>Provides an implementation for the browser viewer, that delegates all but natively browser-supported items to download, when executing the Open action.</td>
</tr>
<tr>
<td>ContentViewer</td>
<td>Provides a widget that contains tabs in which individual viewers for documents are displayed.</td>
</tr>
<tr>
<td>ContentViewPane</td>
<td>Provides an embeddable viewer widget. ContentView creates instances of this widget, to open single viewer instances.</td>
</tr>
<tr>
<td>DocViewer</td>
<td>Provides a base widget for developing custom viewers.</td>
</tr>
<tr>
<td>FilenetViewer</td>
<td>Provides the IBM Daef ViewONE Professional Edition widget.</td>
</tr>
<tr>
<td>HTMLDocViewer</td>
<td>Provides the HTML Conversion Viewer widget.</td>
</tr>
<tr>
<td>IframeDocViewer</td>
<td>Provides a base widget for developing custom viewers that use an iframe to launch. Recommended for Applet based viewers.</td>
</tr>
<tr>
<td>Widget class</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ViewerItem</td>
<td>Represents a viewable item from the repository that is loaded into the viewer framework. This class provides information about the viewer items that is used by the ContentView widget and the ContentViewPane widget.</td>
</tr>
<tr>
<td>PDFDocViewer</td>
<td>Provides the PDF Conversion Viewer widget.</td>
</tr>
</tbody>
</table>

**Parent topic:** [IBM Content Navigator visual widget catalog](#) on page 78
IBM CMIS for Enterprise Content Management API reference

Content Management Interoperability Services (CMIS) is an open source OASIS standard that enables applications to work with one or more content management systems. CMIS defines a standard domain model and standard set of services and protocol bindings for web services and RESTful AtomPub.

The OASIS CMIS Technical Committee maintains a website that includes information about various implementations of CMIS at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cmis.

IBM maintains a page for CMIS Information Management software, which includes links to developerWorks® topics and other useful resources, at http://www.ibm.com/software/data/content-management/cm-interoperability-services.html.

"IBM CMIS for Enterprise Content Management index page"
The IBM CMIS for Enterprise Content Management index page contains information about the instance of IBM CMIS that is deployed to your web application server.

"OASIS CMIS specification" on page 103
The OASIS Content Management Interoperability Services (CMIS) specification includes detailed information about the CMIS Web Services bindings and the RESTful AtomPub bindings.

"IBM CMIS for FileNet Content Manager Development" on page 103
Content Management Interoperability Services (CMIS) is an open source OASIS standard that enables applications to work with one or more content management systems by defining a standard domain model and standard set of services and protocol bindings for Web Services and RESTful AtomPub.

"Programming IBM CMIS for Content Manager applications" on page 116
Content Management Interoperability Services (CMIS) is an open-source OASIS standard that enables applications to work with one or more content management systems by defining a standard domain model and standard set of services and protocol bindings for Web Services and AtomPub.

"IBM CMIS for Content Manager OnDemand implementation of the OASIS CMIS specification" on page 148
IBM CMIS for Content Manager OnDemand includes a limited set of read-only functions from the CMIS specification. IBM CMIS for Content Manager OnDemand does not support creating, updating, or deleting tasks.

IBM CMIS for Enterprise Content Management index page

The IBM CMIS for Enterprise Content Management index page contains information about the instance of IBM CMIS that is deployed to your web application server.

The index page includes links to the service documents and information about the URI templates and service collections that are defined for the repositories on your Enterprise Content Management server. The information depends on the type of IBM CMIS application that is deployed in your environment:

On IBM FileNet P8, the index page describes how to interact with the object stores on your Content Engine server.
On IBM Content Manager Enterprise Edition, the index page describes how to interact with the library servers on your Content Manager EE server.

On IBM Content Manager OnDemand, the index page describes how to interact with the folders on your OnDemand server.

You can use the information about the index page to do these tasks:

- Interact with the OASIS CMIS web services binding or RESTful AtomPub binding
- Develop an application by using an API framework, such as Apache Chemistry API framework. The framework that you use depends on the language in which you develop your application.

When IBM CMIS is properly configured and deployed, you can access the index page at http://host_name:port/context_root/index.jsp. The default context root varies depending on the type of IBM CMIS application that is deployed:

- IBM CMIS for FileNet Content Manager: fnccmis
- IBM CMIS for Content Manager: cmcmis
- IBM CMIS for Content Manager OnDemand: odcmis

The index page also includes links to the web services binding information and the RESTful Atom Protocol (AtomPub) binding information in the OASIS CMIS specification.

**Service collection URLs**

**Root folder collection**

Provides a feed that displays all of the folders and documents that are in the root folder.

**Query collection**

Provides a feed of the objects that satisfy a predefined query.

**Checked out collection**

Provides a feed of the documents that are in the checked-out state in the repository.

**Unfiled collection**

Provides a feed of the documents that are not filed in a folder in the repository.

**Types children collection**

Provides a feed of the base object types, specifically, cmis:document and cmis:folder.

**URI templates**

URI templates enable users to insert values in a standardized URI to retrieve specific data, such as a specific folder or document, from the repository.

The OASIS specification defines the following URI templates:

**Object by ID (objectbyid) template**

Enables users to create a URI that can access a document or folder by specifying the ID of the object.
Object by path (objectbypath) template
   Enables users to create a URI that can access a document or folder by
   specifying the path of the object.

Query (query) template
   Enables users to create a URI to search for data in the repository by
   specifying a CMIS query statement. The server returns a feed of objects
   that correspond to the query that is provided.

Type by ID (typeid) template
   Enables users to create a URI to retrieve metadata about a type (IBM
   FileNet P8 class). The server returns the class description information in
   the CMIS format for a type, which includes all the property descriptions
   that are defined for the specified type.

For more information, see section 3.6.1 URI Templates of the OASIS CMIS
specification.

**OASIS CMIS specification**

The OASIS Content Management Interoperability Services (CMIS) specification
includes detailed information about the CMIS Web Services bindings and the
RESTful AtomPub bindings.

The OASIS CMIS API specification is available in HTML format at
http://docs.oasis-open.org/cmis/CMIS/v1.0/os/cmis-spec-v1.0.html

**Important:** Specific guidance for developing with the OASIS CMIS APIs is not
included in the IBM CMIS for FileNet Content Manager, IBM CMIS for Content
Manager, or IBM CMIS for Content Manager OnDemand documentation. However,
some of the resources in Useful resources provide additional guidance.

**IBM CMIS for FileNet Content Manager Development**

Content Management Interoperability Services (CMIS) is an open source OASIS
standard that enables applications to work with one or more content management
systems by defining a standard domain model and standard set of services and
protocol bindings for Web Services and RESTful AtomPub.

The CMIS Web Services bindings and RESTful AtomPub bindings are described in
the OASIS CMIS specification.
Extensions for IBM CMIS for FileNet Content Manager

The OASIS Content Management Interoperability Services (CMIS) specification enables developers to extend the RESTful AtomPub bindings and Web Services bindings that are provided by the specification. IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) includes several extensions that enable client applications to retrieve additional information from an IBM FileNet Content Manager system.

The IBM CMIS extensions are extensions to the XML output from the server and are present on each response from the Content Engine server. The extensions are optional and do not interfere with client applications that are not designed to use the extensions.

Important: If you design your client application to use the IBM CMIS extensions, it is recommended that you design the application to use the values that are provided by the extensions only if values are available.

In addition, if you design your client application to use the IBM CMIS extensions, you should keep the following restrictions in mind:

- The extensions are for an IBM FileNet Content Manager repository only.
- If your client application is designed to require values for these extensions, your client application will not work with another CMIS-enabled repository.

IBM CMIS for FileNet Content Manager includes the following extensions:

"Class display name extension" on page 105
The class display name (ClassDisplayName) extension enables client applications to retrieve the display name of an object class as part of the object type definition for a collection of objects, such as documents and folders.

"Hidden properties extension" on page 105
The hidden properties (isHidden) extension enables client applications to determine the object properties that should not be displayed based on the object properties that are marked as hidden in Content Engine.

"Choice list symbolic name extension" on page 106
The choice list symbolic name (ChoiceListSymbolicName) extension enables client applications to retrieve the symbolic name of a choice list (also called a controlled vocabulary list or a CVL) from Content Engine.

"Choice list display name extension" on page 106
The choice list display name (ChoiceListDisplayName) extension enables client applications to retrieve the display name of a choice list (also called a controlled vocabulary list or a CVL) from Content Engine.

"Domain name extension" on page 106
The domain name (DomainName) extension enables client applications to retrieve the IBM FileNet P8 domain name as part of the repository information.
The object store ID (ObjectStoreID) extension enables client applications to retrieve the object store GUID as part of the repository information.

The object store database type (ObjectStoreDBType) extension enables client applications to retrieve the type of database, such as DB2® or Oracle, that the repository runs on as part of the repository information.

The records management (RecordsManagementEnabled) extension enables client applications to determine whether the documents in a repository can be declared as records. This information is provided as part of the repository information.

The text search engine (TextSearchEngine) extension enables client applications to determine whether text search is enabled on the repository. If text search is enabled, this extension identifies which text search engine is installed. This information is provided as part of the repository information.

The GZIP compression extension enables client applications to request compressed responses from IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) to increase performance.

Class display name extension
The class display name (ClassDisplayName) extension enables client applications to retrieve the display name of an object class as part of the object type definition for a collection of objects, such as documents and folders.

Remember: If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.

Response format

Hidden properties extension
The hidden properties (isHidden) extension enables client applications to determine the object properties that should not be displayed based on the object properties that are marked as hidden in Content Engine.

Remember: If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.

Response format
**Choice list symbolic name extension**
The choice list symbolic name (ChoiceListSymbolicName) extension enables client applications to retrieve the symbolic name of a choice list (also called a controlled vocabulary list or a CVL) from Content Engine.

**Remember:** If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.

**Response format**

```xml
<p8ext:ChoiceListSymbolicName>Symbolic Name</p8ext:ChoiceListSymbolicName>
```

**Parent topic:** [“Extensions for IBM CMIS for FileNet Content Manager” on page 104](#)

---

**Choice list display name extension**
The choice list display name (ChoiceListDisplayName) extension enables client applications to retrieve the display name of a choice list (also called a controlled vocabulary list or a CVL) from Content Engine.

**Remember:** If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.

**Response format**

```xml
<p8ext:ChoiceListDisplayName>Display Name</p8ext:ChoiceListDisplayName>
```

**Parent topic:** [“Extensions for IBM CMIS for FileNet Content Manager” on page 104](#)

---

**Domain name extension**
The domain name (DomainName) extension enables client applications to retrieve the IBM FileNet P8 domain name as part of the repository information.

**Remember:** If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.

**Response format**

```xml
<p8ext:DomainName>FileNet P8 Domain Name</p8ext:DomainName>
```

**Parent topic:** [“Extensions for IBM CMIS for FileNet Content Manager” on page 104](#)

---

**Object store ID extension**
The object store ID (ObjectStoreID) extension enables client applications to retrieve the object store GUID as part of the repository information.

**Remember:** If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.
Object store database type extension
The object store database type (ObjectStoreDBType) extension enables client applications to retrieve the type of database, such as DB2 or Oracle, that the repository runs on as part of the repository information.

Remember: If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.

Response format
<pext:ObjectStoreDBType>Database type</pext:ObjectStoreDBType>

Records management extension
The records management (RecordsManagementEnabled) extension enables client applications to determine whether the documents in a repository can be declared as records. This information is provided as part of the repository information.

Remember: If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.

Response format
<pext:RecordsManagementEnabled>Boolean true or false</pext:RecordsManagementEnabled>

Text search engine extension
The text search engine (TextSearchEngine) extension enables client applications to determine whether text search is enabled on the repository. If text search is enabled, this extension identifies which text search engine is installed. This information is provided as part of the repository information.

Regardless of the text search engine that is installed, your client must use OASIS CMIS search syntax.

Remember: If you design your client application to use this IBM CMIS extension, it is recommended that you design the application to use the value that is provided by the extension only if a value is available. If this information is not provided, it should not cause your application to encounter an error because the information is not required by the OASIS CMIS API specification.

Response format
<pext:TextSearchEngine>Integer value for text search engine</pext:TextSearchEngine>

The extension returns one of the following values:
The repository does not have a text search engine installed.

1. The repository has an IBM Legacy Content Search Engine text search engine installed.

2. The repository has an IBM Content Search Services text search engine installed.

**Important:** The values that are returned by this extension are documented in the Content Engine API. For definitive information about the text search values, see the `com.filenet.api.constants.CRBSearchType` section of the Content Engine API.

Parent topic: [Extensions for IBM CMIS for FileNet Content Manager](#)

---

**GZIP compression extension**

The GZIP compression extension enables client applications to request compressed responses from IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) to increase performance.

If a client application sends a request that contains an Accept-Encoding header that specifies the GZIP file compression, IBM CMIS compresses the response using GZIP compression.

**Request format**

```
Accept-Encoding: gzip
```

**Response format**

```
Content-Encoding: gzip
```

Parent topic: [Extensions for IBM CMIS for FileNet Content Manager](#)

---

**IBM CMIS for FileNet Content Manager implementation of the OASIS CMIS specification**

The OASIS Content Management Interoperability Services (CMIS) specification enables applications to implement the specification in slightly different ways to account for the differences between repositories. When you develop a client application that uses CMIS services to interact with a repository, you must be aware of the behaviors and limitations of the implementation.

**Tip:** To learn about the optional capabilities that a repository supports, see the values that are published in the `repositoryInfo.capabilities` structure.

In addition, you should be aware of the following behaviors and limitations when you develop a client application to connect to an IBM FileNet Content Manager system by using IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS).

- [Change token support](#)
- [Links in private working copy support](#)

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports the `cmis:changeToken` property for only documents and folders. IBM CMIS does not support the property for domains, relationships, repositories, or content streams.

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports up links in private working copies. However, in some cases the up link leads to an empty feed.
IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports the paging of feeds. However, last links are not supported, and next links sometimes return a page that contains no results.

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports queries. However, there are some restrictions that you should be aware of when developing queries.

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports the `cmis:checkinComment` property.

A CMIS application can retrieve a subset of a content stream by sending an HTTP request that limits the amount of content that is downloaded at one time from the content repository. Limiting the content stream to smaller chunks can help avoid timeouts and other bandwidth problems, particularly when the size of the accessed content is large.

ACL services define a set of operations to retrieve and modify an object's Access Control List (ACL). ACL services are defined in the OASIS Content Management Interoperability Services (CMIS) specification.

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports `cmis:changeToken` property for only documents and folders. IBM CMIS does not support the property for domains, relationships, repositories, or content streams.

The OASIS CMIS specification defines the `cmis:changeToken` property to provide optimistic locking to prevent client applications from updating out-of-date objects.

For more information, see section 2.2.1.3 Change Tokens of the OASIS CMIS specification.

IBM CMIS for FileNet Content Manager supports the `cmis:changeToken` property for only documents and folders.

The `cmis:changeToken` property is included in the object metadata for the `cmis:document` type and the `cmis:folder` type. The `cmis:changeToken` property maps to the IBM FileNet P8 `UpdateSequenceNumber` property.

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports up links in private working copies. However, in some cases the up link leads to an empty feed.
OASIS CMIS specification

The OASIS CMIS specification uses up links that are defined in the AtomPub RFC.

For more information, see section 3.4.3.2 Hierarchy Navigation Internet Draft Link Relations of the OASIS CMIS specification.

IBM CMIS implementation

IBM CMIS for FileNet Content Manager always includes an up link in private working copies, but in some cases the up link leads to an empty feed:

- If a version series includes a version other than the private working copy of the document, the up link in the private working copy leads to an empty feed.
- If a version series has only one version and that version is the private working copy of the document, the up link in the private working copy leads to a feed that contains the parent folder of the private working copy.

Parent topic: IBM CMIS for FileNet Content Manager implementation of the OASIS CMIS specification on page 108

Paging feed support

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports the paging of feeds. However, last links are not supported, and next links sometimes return a page that contains no results.

OASIS CMIS specification

The OASIS CMIS specification uses first, next, previous, and last links that are defined in the AtomPub RFC.

For more information, see section 3.1.8 Paging of Feeds of the OASIS CMIS specification.

IBM CMIS implementation

IBM CMIS for FileNet Content Manager supports the paging of feeds, but has the following behaviors for next and last links:

- Last links
  Last links are not supported.

- Next links
  When the last page of feed is full (more specifically, when the number of items returned equals the value specified for the maxItems argument), IBM CMIS for FileNet Content Manager might provide an unnecessary next link.

Parent topic: IBM CMIS for FileNet Content Manager implementation of the OASIS CMIS specification on page 108

Query support

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports queries. However there are some restrictions that you should be aware of when developing queries.

OASIS CMIS specification

The OASIS CMIS specification provides a type-based query service to search for objects on the repository.
For more information, see section 2.1.10 Query of the OASIS CMIS specification.

IBM CMIS implementation

IBM CMIS for FileNet Content Manager supports the OASIS CMIS query specifications with the following exceptions and limitations.

Character escaping in LIKE strings is not supported

The OASIS CMIS specification requires repositories to support the backslash (\) as an escape character in query statements. One of the requirements is that in a LIKE string, \% represents % and \_ represents _.

However, IBM CMIS for FileNet Content Manager does not support querying on the percent sign (%) or the underscore (_) characters in a LIKE string because the Content Engine APIs do not support ESCAPE clauses. If a query includes an escaped percent sign (\%) or an escaped underscore (\_), IBM CMIS for FileNet Content Manager converts the characters to an underscore, which is a single-character wildcard.

Converting escaped percent signs and escaped underscores to a single-character wildcard character can result in more search results than expected.

For example, you have three documents in your repository named document_1, document_2, and document11. You use the query SELECT * FROM Document WHERE DocumentTitle LIKE 'document\_%' to search for all documents that have a title that starts with document_. The expected result is that document_1 and document_2 are returned by the query.

However, when you submit the query, IBM CMIS for FileNet Content Manager converts the query to (SELECT * FROM Document WHERE DocumentTitle LIKE 'document_%'), so that IBM FileNet P8 can parse the query. This query returns all three documents.

Tip: You can use percent signs and underscores as wildcard characters in queries that include equality conditions because equality conditions do not require escape characters. For example, you can use the query SELECT * FROM Document WHERE DocumentTitle = 'document_1' OR DocumentTitle = 'document_2' to return document_1 and document_2.

Parentheses are required for nested JOIN clauses

The OASIS CMIS specification defines the following syntax for a query that contains nested JOIN clauses:

<joined table> ::= "(" <joined table> ")" |
| <table reference> [ <join type> ] JOIN <table reference>

In the OASIS CMIS specification, the parentheses around the nested <joined table> are optional. However, IBM CMIS for FileNet Content Manager requires parentheses around the nested <joined table>.

For example, if you submit the following query, IBM FileNet P8 returns a parsing error on the INNER JOIN clause:

```sql
SELECT d.* FROM cmis:document d INNER JOIN cmis:folder f ON d.cmis:createdBy = f.cmis:createdBy INNER JOIN EntryTemplate e ON d.cmis:lastModifiedBy = e.cmis:lastModifiedBy WHERE e.cmis:createdBy<>'admin'
```
To run this query in IBM CMIS for FileNet Content Manager, you must use the following syntax:

```sql
SELECT d.* FROM (cmis:document d INNER JOIN cmis:folder f ON
d.cmis:createdBy = f.cmis:createdBy) INNER JOIN EntryTemplate e ON
d.cmis:lastModifiedBy = e.cmis:lastModifiedBy WHERE
e.cmis:createdBy='admin'
```

**Nested JOIN clauses can appear on only one side of the JOIN keyword**

The OASIS CMIS specification defines the following syntax for a query that contains nested JOIN clauses on both sides of the JOIN keyword:

```sql
<table reference> ::= <table name> [ [ AS ] <correlation name> ]
| <joined table>

<joined table> ::= "(" <joined table> ")"
| <table reference> [ <join type> ] JOIN <table reference>

<join specification>
```

In the OASIS CMIS specification, you can have nested JOIN clauses on either side of the JOIN keyword. However, IBM CMIS for FileNet Content Manager supports nested JOIN clauses on either the left side or the right side of the JOIN keyword, but not on both sides of the JOIN keyword.

For example, if you submit the following query, IBM FileNet P8 returns a parsing error because there is a nested JOIN clause on each side of the JOIN keyword:

```sql
SELECT d.* FROM (cmis:document d INNER JOIN cmis:folder f ON
d.cmis:createdBy = f.cmis:createdBy) INNER JOIN (EntryTemplate
e INNER JOIN Email e2 ON e.cmis:createdBy = e2.cmis:createdBy)
ON d.cmis:lastModifiedBy = e.cmis:lastModifiedBy WHERE
e.cmis:createdBy='admin'
```

To run this query in IBM CMIS for FileNet Content Manager, you must remove the nested JOIN clause on one side of the JOIN keyword. The following query has a nested JOIN clause on the right side of the JOIN keyword:

```sql
SELECT d.* FROM EntryTemplate e INNER JOIN (cmis:document d
INNER JOIN cmis:folder f ON d.cmis:createdBy = f.cmis:createdBy) ON
d.cmis:lastModifiedBy = e.cmis:lastModifiedBy WHERE
e.cmis:createdBy='admin'
```

**Versioning properties on document objects support**

IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) supports the `cmis:checkinComment` property.

**OASIS CMIS specification**

The OASIS CMIS specification defines a set of versioning properties for document objects, which includes the `cmis:checkinComment` property.

For more information, see section 2.1.9.5 Versioning Properties on Document Objects of the OASIS CMIS specification.

**IBM CMIS implementation**

When a value is specified for the IBM CMIS `checkinCommentProperty` parameter, the `cmis:checkinComment` setting persists in the Content Engine repository. The `checkinCommentProperty` value is a symbolic name for an existing Content Engine
string property in the same document class or subclass of the object store where the document is stored.

Parent topic: “IBM CMIS for FileNet Content Manager implementation of the OASIS CMIS specification” on page 108

Partial content stream retrieval in HTTP requests
A CMIS application can retrieve a subset of a content stream by sending an HTTP request that limits the amount of content that is downloaded at one time from the content repository. Limiting the content stream to smaller chunks can help avoid timeouts and other bandwidth problems, particularly when the size of the accessed content is large.

OASIS CMIS specification
The OASIS CMIS specification states that specifying a byte range in an HTTP request for a content stream is a valid method for limiting the amount of content retrieval at one time.

IBM CMIS implementation
IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS) implements a subset of the HTTP range request header functionality documented in OASIS and RFC specifications. IBM CMIS does not support multiple-byte ranges in the HTTP range request header. If more than one byte range is in the HTTP range request header, the content stream contains only the content that is associated with the first byte range.

For example, if an HTTP request specifies the following byte ranges:

Range: bytes=500-999, 1300-1700

then the HTTP response contains only bytes 500 - 999 of the content stream. The second range (1300 - 1700) does not generate an error, but no content in these parameters is returned because it is a secondary range in the header.

Parent topic: “IBM CMIS for FileNet Content Manager implementation of the OASIS CMIS specification” on page 108

Permission mappings in ACL services
ACL services define a set of operations to retrieve and modify an object’s Access Control List (ACL). ACL services are defined in the OASIS Content Management Interoperability Services (CMIS) specification.

CMIS ACL and Permission Definitions
By CMIS specification, an ACL is a list of Access Control Entries (ACEs). For more information, see http://docs.oasis-open.org/cmis/CMIS/v1.0/os/cmis-spec-v1.0.html#_Toc243905406

An ACE contains the following components:
- One Principal: A principal represents a user management object, for example, a user, group, or role. A principal contains one String with the principalid element.
- One or more Strings with the names of the permissions.
- A Boolean flag direct, where TRUE indicates that the ACE is directly assigned to the object. If FALSE, that the ACE is somehow derived.

The CMIS specification defines three basic permissions:
<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmis:read</td>
<td>Permissions to read the content.</td>
</tr>
<tr>
<td>cmis:write</td>
<td>Permissions of cmis:read and the permissions to edit the content.</td>
</tr>
<tr>
<td>cmis:all</td>
<td>This permission is used to express all of the permissions of a repository and includes cmis:read and cmis:write.</td>
</tr>
</tbody>
</table>

The specification indicates that repositories can extend this set with repository-specific permissions.

**Permissions in CMIS and FileNet P8**

The security of a FileNet P8 Content Engine object is expressed by access rights. For more information, see the information in the Object permissions section:

[http://www.ibm.com/support/knowledgecenter/api/content/SSNW2F_5.2.0/com.ibm.p8.ce.dev.ce.doc/sec_concepts.htm#sec_concepts__sec_basic_object_permissions](http://www.ibm.com/support/knowledgecenter/api/content/SSNW2F_5.2.0/com.ibm.p8.ce.dev.ce.doc/sec_concepts.htm#sec_concepts__sec_basic_object_permissions)

To convey the native security settings of FileNet P8 Content Engine objects to CMIS compatible client applications without losing information and while supporting the CMIS specification better, IBM CMIS for FileNet Content Manager uses a mixed approach to map between a FileNet P8 object's security settings in Content Engine and CMIS object's permissions.

The permission strings in a CMIS ACE can have CMIS basic permission strings, FileNet P8 AccessRight constants, or both. For backward compatibility support, IBM CMIS for FileNet Content Manager still allows FileNet P8 AccessLevel constants in a CMIS object's permissions (AccessLevel is a deprecated constant in FileNet P8 Version 5.2 API).

The permission mapping occurs in both directions between IBM CMIS for FileNet Content Manager and CMIS compatible client applications. The mapping is controlled by a set of configuration parameters in the IBM CMIS for FileNet Content Manager configuration file, cmis.properties. The parameters define what FileNet P8 access rights are contained in a CMIS basic permission. The CMIS basic permission mappings are configured for document, folder, and other objects separately because the same CMIS basic permission can mean different FileNet P8 access rights for different base object types. For example, consider the following mapping:

```
CmisToP8PermissionMapp.Document.Read = AccessLevel.READ | AccessRight.VIEW_CONTENT
```

This mapping means that cmis:read for document type corresponds to the FileNet P8 access level READ AND the access right VIEW_CONTENT.

**Downstream permission mapping**

The permission mapping from FileNet P8 access rights to permissions received by CMIS compatible client applications is computed by using the following algorithm:

1. A FileNet P8 grantee's access rights are attempted to be mapped to cmis:all by the mapping definitions in cmis.properties. If the grantee's access rights are equal or greater than the defined mapping for cmis:all, then cmis:all is added to the permission strings of the grantee (which is called the principal in the CMIS specification). If the grantee's access rights are less than the defined
mapping for cmis:all, they are attempted to be mapped to cmis:write; and then cmis:read. Then, the mapped access rights are removed from the grantee's original access rights.

2. Following step 1, if the grantee's access rights cannot be mapped to cmis:all, cmis:write or cmis:read exactly, the remainder of the access rights are added into the permission strings of the grantee. The string representation of the permission will become each access right constant in FileNet P8 API with the following prefix: fn.

For example, if the cmis.properties mapping file has a mapping definition,

\[
\text{CmisToP8PermissionMapp.Document.Read = AccessLevel.READ | AccessRight.VIEW_CONTENT,}
\]

and a grantee's access rights are AccessRight.READ, AccessRight.READ_ACL, AccessRight.VIEW_CONTENT, and AccessRight.MINOR_VERSION, the permission mapping result will be cmis:read, fn:MINOR_VERSION in the response to the CMIScompatible client applications.

**Upstream permission mapping**

The permission mapping from CMIS compatible clients to FileNet P8 access rights is computed by the following algorithm:

1. Only CMIS basic permissions (cmis:all, cmis:write, and cmis:read) and FileNet P8 AccessRight and AccessLevel constants with the fn prefix are allowed.

2. CMIS basic permissions are mapped to FileNet P8 access rights according to the mapping definitions in the cmis.properties mapping file.

3. Prefixes are moved from AccessRight and AccessLevel constants with the fn prefix and they are treated as they are: as access rights and access levels.

4. Access rights from step 2 and step 3 are combined together (bit operation OR) to be come the new access rights of the object in the context.

For example, assume that the cmis.properties mapping file has the following mapping definition:

\[
\text{CmisToP8PermissionMapp.Document.Read = AccessLevel.READ | AccessRight.VIEW_CONTENT}
\]

If the client requests to modify an object's permissions for a principal to cmis:read, fn.MINOR_VERSION, then the principal's access rights to the object in Content Engine will be changed to AccessRight.READ | AccessRight.READ_ACL | AccessRight.VIEW_CONTENT | AccessRight.MINOR_VERSION.

**Best practices for developing client applications**

There are several best practices and recommendations that you should keep in mind when accessing content through IBM Content Management Interoperability Services for FileNet Content Manager (IBM CMIS).

**Recommendations for using the deleteTree service**

The deleteTree service deletes a folder and all of its subfolders and documents.

If you call the deleteTree service with the continueOnFailure parameter set to true, folders and documents are deleted individually. If a document or folder cannot be deleted, the service moves to the next document or folder in the list. When the service completes, it returns a list of the document IDs and folder IDs that were not deleted to the client application.
If you call the `deleteTree` service with the `continueOnFailure` parameter set to false, all of the folders and documents are deleted in a single batch, which improves performance. However, if a document or folder cannot be deleted, none of the contents of the folder are deleted and an error is returned to the client.

When `continueOnFailure` is set to false, the service can time out if the parent folder contains many documents and folders. If this occurs, you can call the `deleteTree` service on a subfolder of the parent folder to reduce the number of objects that are deleted at one time and to avoid a timeout.

**Recommendations for getting unfiled documents**

When you retrieve unfiled documents by using the Unfiled service collection URL, the service can consume system resources in a large system. If you implement the following suggestions, you can reduce the impact of using the Unfiled service collection URL:

- In REST feeds, use next links to page through results rather than requesting the next page with a skip count.
- If you want to retrieve a large number of unfiled documents, specify a larger value for the `maxItems` parameter to reduce the amount of paging that is required to display the documents.
  This is especially important if you use Web Services to retrieve documents because Web Services does not provide a next link.
- If you want to retrieve a specific set of unfiled documents, use a custom query to return only the wanted documents.
- If your repository (IBM FileNet P8 object store) contains more than 1,000,000 documents, use the Unfiled service collection URL only when the least number of users are using the system.

**Recommendations for working with large objects**

If your repository contains large objects, you should implement the following recommendations to reduce memory consumption:

- For REST Services, the most efficient method to access large objects is the `setContentStream` method. The `setContentStream` method is defined in section 2.2.4.16 `setContentStream` of the OASIS CMIS API specification.
- For Web Services, the most efficient method to access large objects is the SOAP Message Transmission Optimization Mechanism (MTOM) method, which must be configured in client application.

Parent topic: IBM CMIS for FileNet Content Manager Development” on page 103
The CMIS Web Services bindings and RESTful AtomPub bindings are described in the OASIS CMIS specification.

Information about CMIS client APIs is provided by the vendors of the client APIs.

**Extension for IBM CMIS for Content Manager**

The OASIS Content Management Interoperability Services (CMIS) specification enables developers to extend the RESTful AtomPub bindings and Web Services bindings that are provided by the specification. IBM Content Management Interoperability Services for Content Manager (IBM CMIS) includes an extension that enable client applications to retrieve additional information from a Content Manager EE system.

**IBM CMIS for Content Manager implementation of the OASIS CMIS specification** on page 118

The OASIS Content Management Interoperability Services (CMIS) specification can be implemented in different ways. When you develop a client application that uses CMIS services to interact with a repository, you need to be aware of the behaviors and limitations of the implementation.

**Parent topic:** “IBM CMIS for Enterprise Content Management API reference” on page 101

---

**Extension for IBM CMIS for Content Manager**

The OASIS Content Management Interoperability Services (CMIS) specification enables developers to extend the RESTful AtomPub bindings and Web Services bindings that are provided by the specification. IBM Content Management Interoperability Services for Content Manager (IBM CMIS) includes an extension that enable client applications to retrieve additional information from a Content Manager EE system.

The IBM CMIS extension is an extension to the XML output from the server and are present on each response from a Content Manager EE server. The extension is optional and does not interfere with client applications that are not designed to use the extension.

**Important:** If you design your client application to use the IBM CMIS extension, it is recommended that you design the application to use the values that are provided by the extension only if values are available.

In addition, if you design your client application to use the IBM CMIS extension, you should keep the following restrictions in mind:

- The extension is for a Content Manager EE repository only.
- If your client application is designed to require values for the extension, your client application will not work with another OASIS CMIS-enabled repository.

IBM CMIS for Content Manager includes the following extension:

**GZIP compression extension** on page 118

The GZIP compression extension enables client applications to request compressed responses from IBM Content Management Interoperability Services for Content Manager (IBM CMIS) to increase performance.

**Parent topic:** “Programming IBM CMIS for Content Manager applications” on page 116
**GZIP compression extension**

The GZIP compression extension enables client applications to request compressed responses from IBM Content Management Interoperability Services for Content Manager (IBM CMIS) to increase performance.

If a client application sends a request that contains an Accept-Encoding header that specifies the GZIP file compression, IBM CMIS compresses the response as a GZIP file.

**Request format**

Accept-Encoding: gzip

**Response format**

Content-Encoding: gzip

Parent topic: "Extension for IBM CMIS for Content Manager” on page 117

---

**IBM CMIS for Content Manager implementation of the OASIS CMIS specification**

The OASIS Content Management Interoperability Services (CMIS) specification can be implemented in different ways. When you develop a client application that uses CMIS services to interact with a repository, you need to be aware of the behaviors and limitations of the implementation.

You should be aware of the following behaviors and limitations when you develop a client application to connect to a Content Manager EE system by using IBM Content Management Interoperability Services for Content Manager (IBM CMIS).

"Links in private working copy support”

IBM Content Management Interoperability Services for Content Manager (IBM CMIS) supports up links in private working copies. However, in some cases the up link leads to an empty feed.

"Paging feed support” on page 119

IBM Content Management Interoperability Services for Content Manager (IBM CMIS) supports the paging of feeds. However, last links are not supported.

"Query support” on page 119

IBM Content Management Interoperability Services for Content Manager (IBM CMIS) supports queries. However there are some restrictions that you should be aware of when developing queries.

"Working with documents” on page 119

IBM CMIS for Content Manager enables you to create, search, or delete folders and their contents. It also provides result set paging and document versioning capabilities.

Parent topic: "Programming IBM CMIS for Content Manager applications” on page 116

---

**Links in private working copy support**

IBM Content Management Interoperability Services for Content Manager (IBM CMIS) supports up links in private working copies. However, in some cases the up link leads to an empty feed.

**OASIS CMIS specification**

The OASIS CMIS specification (CMIS Version 1.0 Committee Specification 01) leverages up links that are defined in the AtomPub RFC.
For more information, see section 3.4.3.2 Hierarchy Navigation Internet Draft Link Relations of the OASIS CMIS specification.

IBM CMIS implementation

IBM CMIS for Content Manager always includes an up link in private working copies, but in some cases the up link leads to an empty feed:

- If a version series includes a version other than the private working copy of the document, the up link in the private working copy leads to an empty feed.
- If a version series has only one version and that version is the private working copy of the document, the up link in the private working copy leads to a feed that contains the parent folder of the private working copy.

Parent topic: “IBM CMIS for Content Manager implementation of the OASIS CMIS specification” on page 118

Paging feed support

IBM Content Management Interoperability Services for Content Manager (IBM CMIS) supports the paging of feeds. However, last links are not supported.

OASIS CMIS specification

The OASIS CMIS specification (CMIS Version 1.0 Committee Specification 01) leverages first, next, previous, and last links that are defined in the AtomPub RFC.

For more information, see section 3.1.8 Paging of Feeds of the OASIS CMIS specification.

IBM CMIS implementation

IBM CMIS for Content Manager supports the paging of feeds, but has the following behaviors for last links:

Last links

Last links are not supported.

Parent topic: “IBM CMIS for Content Manager implementation of the OASIS CMIS specification” on page 118

Query support

IBM Content Management Interoperability Services for Content Manager (IBM CMIS) supports queries. However there are some restrictions that you should be aware of when developing queries.

OASIS CMIS specification

The OASIS CMIS specification (CMIS Version 1.0 Committee Specification 01) provides a type-based query service to search for objects on the repository.

For more information, see section 2.1.10 Query of the OASIS CMIS specification.

Parent topic: “IBM CMIS for Content Manager implementation of the OASIS CMIS specification” on page 118

Working with documents

IBM CMIS for Content Manager enables you to create, search, or delete folders and their contents. It also provides result set paging and document versioning capabilities.
IBM CMIS for Content Manager enables you to publish or republish documents, delete folders and subfolders within a folder, and map document types and property sheets of client applications to item types and attribute groups.

### Labels, file names

IBM CMIS for Content Manager supports up to 255 characters for labels (file names) for client applications.

### Item types

The server can contain many item types, including item types that are used by other applications that interoperate on the same repository. You must ensure that you configure your system so that IBM CMIS for Content Manager users select the appropriate item type when adding documents in .

### Folder types

IBM CMIS for Content Manager uses the designated folder types or default folder type ClbFolder. to create new folders. The ClbFolder item type is a CMIS-optimized item type.

### Document types optimized for CMIS

In IBM CMIS for Content Manager, a document is known as a converted, extended, or CMIS-optimized document type if it has extended metadata for the standardized name attribute and databases indexes. CMIS-optimized document types support optional extended metadata that adds features and optimizes performance with IBM CMIS for Content Manager.

### Model mapping

IBM CMIS for Content Manager maps IBM CMIS object types to Content Manager EE item types.

### Permission mappings in ACL services

ACL services define a set of operations to retrieve and modify an object’s Access Control List (ACL). ACL services are defined in the OASIS Content Management Interoperability Services (CMIS) specification.

### Paging

Paging enables you to view subsets of results, one page at a time, instead of retrieving and scrolling through the full result set. A page of results contains a subset of items from a potentially larger set of results.

### Deleting folders

When you delete a folder, the deletion typically cascades to the folder contents too. All subfolders and documents that are visible to you are deleted along with the folder.

### Versioning

Versioning is the ability to store and maintain multiple versions of an item. You specify versioning rules when you define an item type. If an item type is enabled for versioning, all items in that item type can be versioned manually or automatically depending on the version control settings and the application support for versioning.

---

**Parent topic:** IBM CMIS for Content Manager implementation of the OASIS CMIS specification

**Labels, file names:**

IBM CMIS for Content Manager supports up to 255 characters for labels (file names) for client applications.

Unconverted item types are limited by the length defined for the particular attribute mapped for a file name based on the particular item type. The mapped
name attribute can be the OriginalFileName field on resource items and document parts, representative attribute, or other custom attribute. The repository is limited only by the attribute definitions in the repository that store the file name.

Applications and operating systems might impose different limits affecting only that application or operating system. The application is responsible for handling file name lengths that it does not support or are not supported on the particular operating system, including handling existing items that are created by other applications without the limitation.

"Character set limitations for labels (file names)"
IBM CMIS for Content Manager imposes only one character set limitation for labels, which are usually used as file names and appear in path segments.

"Multiple items with the same file name"
Content Manager EE supports multiple items that use the same file name or label within the same parent folder.

Parent topic: "Working with documents” on page 119

Character set limitations for labels (file names):

IBM CMIS for Content Manager imposes only one character set limitation for labels, which are usually used as file names and appear in path segments.

Some applications impose additional character set limitations when creating new content based on restrictions for the particular application or operating system. Therefore, you might find additional limitations imposed by the application, not the repository. However, multiple applications operate on the same repository and each can have different character set restrictions or none at all. Each application is responsible for encoding and decoding characters that are not supported by the particular application when working with data created by other applications.

Parent topic: "Labels, file names” on page 120

Multiple items with the same file name:

Content Manager EE supports multiple items that use the same file name or label within the same parent folder.

Depending on the application, the application might check for existence of an item before creating an item. The application might report an application error or prompt the user to replace the item if one with same name exists. The repository allows multiple items in the same folder to use the same name depending on whether the item type configuration is hierarchical or not. IBM CMIS for Content Manager adds unique name enforcement for all item types within a folder regardless of item type configuration if path indexing is enabled. The data created or modified by other applications, created without path indexing, or existing data can have the same names when not restricted by the repository. IBM CMIS for Content Manager supports existing data in the repository with the same name by generating a unique path to each item with the same name that works with most applications. When path indexing is enabled, the existing name value is reported with a slightly altered name and is saved if the document or folder is modified.

For repository data found with same name values, the name and path are reported slightly differently depending on the following conditions:

• If path indexing is enabled and the data is indexed, the name and path include sequential copy number in parentheses.
• If not indexed and fast paths are enabled, the name is unchanged but the path value includes an embedded ID.

• If not indexed, fast paths are disabled, and item type and data use the configurable ideal name attribute, the name and path are unchanged and use the same value. However, the ambiguous path reports an error if used.

Indexed path

When path indexing is enabled, existing data and data from other applications that is indexed by the path index crawler are supported by adding a parentheses with a sequential copy number to the file name or folder name. When folder operations or queries are optimized by path indexing, the results are filtered or sorted according to the augmented name and path value instead of the original name. When you save the changes, the augmented name value is saved to the document or folder as the new name value. The users can still change the name to any unique value. When you create or modify documents or folders through IBM CMIS for Content Manager, duplicate names are not allowed.

Fast paths

For fast paths, IBM CMIS for Content Manager generates a unique ID and unique path for each item.

The fast access paths that are generated by using composite names (rather than just the non-unique label) provide for a unique path, but the fast path still reports the label (file name) and title as they were submitted for the application to show to the user. If you disable fast paths, you do not get the performance and scaling benefits of fast paths. In addition, if you disable fast paths, the generated paths might not be unique because they use only labels (names) instead of the composite information. Accessing items by a non-unique path results in an error if the unrevised path resolves to more than one matching item.

Ambiguous paths

With fast access paths and path indexing disabled, the repository supports creation of multiple items with the same label. However, the application might detect the existing item by path before creating one with the same name and instead prompt the user.

Two users with different permissions might be able to create items with the same name and not have a problem. However, a third user who has permissions to see both items cannot access them by a non-unique path unless unique fast access paths are enabled.

If you try to open an item when multiple items exist in the same parent folder with the same name, then depending on the application, you might receive an error reporting that multiple items exist with the same name and the path submitted cannot uniquely identify which one of the same-named items you want to open. If the application uses the actual path returned for each individual item, including composite names, then the path uniquely identifies which item you selected. If you receive this error, refresh your folder view. Then, retrieve the folder contents list with valid unique paths for each item.

For example, with some client applications, you can create multiple copies of the same folder if you create or drag the same named folder multiple times. However, you cannot open any of the new copies until you refresh your folder view because
the application does not track the path that uniquely identifies the item until you refresh.

**Differences in application behavior**

As mentioned previously, not all applications handle same-named or same labeled items within the same folder. For example, the application might depend on the label (or file name) field rather than the generated unique path and ID. Applications are expected to gracefully handle multiple items with the same name or label, so that the application does not fail. It is up to the application to determine its handling. Some applications might show of the all same-named items and internally map them by using the unique path or ID. Other applications might filter out duplicate names and show only one. If the application accesses the remaining items by ID or true unique path, you might be able to use the items. However, if the application attempts to access them without using the generated unique path or ID, you get an error indicating that the path does not uniquely identify a single item.

For example, some client applications handle same-named items that exist, and allow you to access each separately.

**Item types:**

The server can contain many item types, including item types that are used by other applications that interoperate on the same repository. You must ensure that you configure your system so that IBM CMIS for Content Manager users select the appropriate item type when adding documents in .

- **”Item type classifications” on page 124**
  For best performance and behavior optimized for normal documents, use item types with item type classification resource. Item types with the classification of resource are optimized for a single binary resource and work best for normal documents.

- **”Special purpose item types” on page 124**
  Some item types are created for specific purposes by applications and might not be intended for creating and saving documents, such as ICMFORMS and ICMSAVEDSEARCH.

- **”Hierarchical item types and folder item types” on page 125**
  Users can create, browse, and edit documents and folders whose types are marked by the administrator as hierarchical. Administrators can mount folders from the hierarchical model or the hierarchical root itself for displaying and browsing within IBM CMIS.

- **”Folders-only item types” on page 126**
  Item types with a folder-only option that is enabled can contain only folders. This is a Content Manager EE restriction.

- **”Item type limitations for Content Manager for z/OS” on page 127**
  If your repository is Content Manager for z/OS, the maximum number of item types available to users must be limited to 19 item types by default.

**Related concepts:**
- **”Document type and folder type mapping” on page 135**
  CMIS object types are the combined set of document types and folder types. CMIS object types are mapped directly to Content Manager EE item types.
Item type classifications:

For best performance and behavior optimized for normal documents, use item types with item type classification resource. Item types with the classification of resource are optimized for a single binary resource and work best for normal documents.

However, you can use item types with item type classification document for normal documents. Item types with the classification of document support a compound document model with multiple binary content elements in Content Manager EE. The item type is not optimized for normal documents with a single binary resource. Operations like browsing documents and opening documents take slightly more time to obtain data about for the binary content element. Obtaining the binary content element metadata such as name, size, and MIME type takes three repository calls for an item type with classification document. But obtaining the binary content element metadata for an item type with classification resource takes only one repository call. However, even with three repository calls, document access should still be fast, but just not as optimal as a single repository call. Additionally, some data on the binary content elements are not always retrieved in a document listing, such as file size. Therefore when browsing folders, your document size might indicate 0 bytes until you select and retrieve the individual document.

You can use item types with item type classification item, but are restricted to metadata only documents. Item types with the classification of item support metadata only and zero binary content elements. The maximum file size is zero. If you open the binary content of a document that supports metadata only, you see a blank document. If you save binary content with more than 0 bytes, you get an error message.

Parent topic: “Item types” on page 123
Related concepts:
"Document type and folder type mapping” on page 135
CMIS object types are the combined set of document types and folder types. CMIS object types are mapped directly to Content Manager EE item types.

Special purpose item types:

Some item types are created for specific purposes by applications and might not be intended for creating and saving documents, such as ICMFORMS and ICMSAVEDSEARCH.

Specifically, do not allow users to use the ICMFORMS and ICMSAVEDSEARCH item types that ship with to create documents. When you configure security for your item types, grant or restrict user permissions so that users see only valid item types when they view or create documents. For users that use multiple applications, some of which have access to the specialized item types, the users must rely on the item type names and descriptions to guide them in using the appropriate item types to create documents.

Some item types might be used by applications for document organization, custom folder types, pieces of a compound document model, or to save data for specific features, such as ICMSAVEDSEARCH or ICMFORMS. Item types do not have any flags for an application to generically detect the intended use other than by name or description. Therefore all independent item types appear as document types. It is up to the administrator to grant user permissions to view or create items using any item type, define the appropriate classification, and set any appropriate constraints on metadata to ensure the fields are entered. If users have access to view
specialized item types and create items using them, and you do not want to restrict access to create items for the item type, educate your users on which document types you recommend that they use to ensure that they do not create items under certain item types. Some users can create items under some specialized item types so that they can use features in other applications such as saved searches or forms. Users that have access to specialized item types should read the name or description and use the appropriate document types to create items.

Your Content Manager EE server might contain a predefined item type named ICMFORMS. The ICMFORMS item type comes from the Content Manager EE, not IBM CMIS for Content Manager. The ICMFORMS item type has an incomplete configuration. The configuration for ICMFORMS is incomplete because the default resource manager and SMS Collection codes are set to the invalid value of zero. If you attempt to create items using the native APIs, or indirectly through service APIs, the Content Manager EE APIs report an internal error because the item type definition is incomplete due to invalid values. An administrator must complete the ICMFORMS item type configuration before using it to create items.

**Recommendation:** Do not use ICMFORMS as a document type. Instead, you should use the document types that are configured for storing documents, not forms.

**Parent topic:** “Item types” on page 123

Hierarchical item types and folder item types:

Users can create, browse, and edit documents and folders whose types are marked by the administrator as hierarchical. Administrators can mount folders from the hierarchical model or the hierarchical root itself for displaying and browsing within IBM CMIS.

In Content Manager EE Version 8.4.3, you create a hierarchical item type by enabling the hierarchical option when an item type is created. When the hierarchical option of an item type is enabled, it enforces hierarchical constraints for its items and causes hierarchical feature-related metadata to be added.

Users can see and browse documents and folders from hierarchical types and folders that descend from the repository hierarchical root. The benefits of using hierarchical item types is unique file names within parent folders. Administrators can mark folder types directly in Content Manager EE that are interoperable with all Content Manager EE applications and understood as folder types, not just IBM CMIS, and enforced by the Content Manager EE server. The folder types and document types does not get mixed up between different Content Manager EE applications. Users can depend on documents and folders always filed in a hierarchy by all applications in a more interoperable way without the need for an administrator to search out and mount folder trees created by other applications.

The DocumentTypeEnable tool supports creating new item types and supports the option to make it a hierarchical item type or not.

After an item type has been created, the hierarchical option cannot be changed. Items of a hierarchical item type (referred to as hierarchical items) have the following characteristics:

- A single root folder exists for all hierarchical items called the hierarchical root.
- Every item that is of a hierarchical item type must have a name value (ICM$NAME).
• A hierarchical item (including folders) must have a single hierarchical parent folder (except for the hierarchical root folder).

• A hierarchical item (including folders) can always be reached from the root hierarchical folder.

• Within a given hierarchical parent folder, each hierarchical item name must be unique.

• A hierarchical parent folder can be deleted only if it is empty.

• Hierarchical item types with the folders-only option set to true can hold only hierarchical folder items (semantic type equals Folder).

Hierarchical folder item types

A hierarchical item type is considered to be a hierarchical folder item type if its folders-only option is enabled. A hierarchical folder item type has all the characteristics of a hierarchical item type and obeys all hierarchical constraints. In addition, all of its hierarchical folder items must be of the Folder semantic type. A hierarchical item type contains non-folder items and documents only while a hierarchical folder item type only contains folders. Hierarchical folders can be created only in hierarchical item types that are configured with the folders-only option. Conversely, all non-folder hierarchical items must be created in an item type that has the folders-only option disabled. The folders-only option of a hierarchical folder item type cannot be disabled if there are existing hierarchical folders.

Therefore, in IBM CMIS, when you create documents and folders of hierarchical item type or hierarchical folder item type, the documents and folders that you create are constrained by the characteristics of the hierarchical item types. For example, if you try to move an hierarchical document from a hierarchical folder to a non-hierarchical folder, you get an error message. Therefore, you can only move and file hierarchical type document types and folder types under an existing hierarchical folder or hierarchical root, etc.

Parent topic: Item types on page 123

Folders-only item types:

Item types with a folder-only option that is enabled can contain only folders. This is a Content Manager EE restriction.

Content Manager EE Version 8.4.3 or later supports identifying new item types as folders only. Any item type defined in Content Manager EE with the folders-only option set to TRUE is restricted by the Content Manager EE repository to allow only folders to be created by any application and cannot be overridden. All item types defined with the folders-only option are automatically identified as folder types and are not required to be listed in the folderTypes configuration setting of IBM CMIS. For item types with folders-only option set to FALSE, the behavior of the item type depends on whether the item type is hierarchical or not.

Content Manager EE Version 8.4.3 or later also supports identifying new item types as hierarchical item types. The hierarchical item types add new semantics that does not allow an item type to be used for both folders and documents by any application. Any hierarchical item type that is not marked as folders only are restricted to documents and cannot be used to create folders by any application.

The DocumentTypeEnable tool supports creating new item types and supports the option to mark it as folders only or not.
If the item type is hierarchical and the folders-only option is set to FALSE, the item type is used for document types only and is used to create documents, not folders. If the item type is not hierarchical and the folders-only option is set to FALSE, the item type is considered as a document type by IBM CMIS unless this item type is listed as a folder type in the `folderTypes` configuration setting. If the item type is listed as a folder type by using the `folderTypes` configuration setting, IBM CMIS considers this item type as a folder type. You can continue using the item type that is not hierarchical and not folders-only for either documents or folders, but IBM CMIS considers this item type as a document type only. However, IBM CMIS works with folders created by this item type by other applications.

**Parent topic:** [“Item types” on page 123](#)

### Item type limitations for Content Manager for z/OS:

If your repository is Content Manager for z/OS, the maximum number of item types available to users must be limited to 19 item types by default.

If there are too many item types defined in the repository that are accessible to users, operations such as opening a library or folder and running a saved search fail. The number of item types that cause an error vary by feature, configuration settings, and by application. For example, the applications that display paged and sorted results are more likely to cause this error with fewer item types than an application that displays all results and sorts within the application. Opening a library or folder is more likely to cause this error with fewer item types than running a saved search with a simple query. The lowest maximum number of item types before the first feature usually exceeds the maximum with default configuration settings is 19 item types. However, running a saved search that is defined with a complex query can have a lower maximum number of item types than 19 and varies by complexity of the query. An unusual and extremely complex query might be too complex to run. However, most saved searches are not very complex and can usually support more than the maximum supported by opening a library or folder.

The limitation applies to the number of item types in the system, not just document types and even if hidden in document type listings. For example, the count includes the IBM CMIS for Content Manager model item types, such as ClbLibrary, ClbApplication, ClbSavedSearch, and ClbFolder that are required for features. The count also includes item types for compound document parts, such as ICMBASE, ICMBASETTEXT, ICMBASESTREAM, ICMNOTELOG, etc.

The maximum of 19 is the count of all normal item types that you see in the Content Manager EE system administration client under **Item types** in the data modeling section. The list includes all predefined non-system item types and user defined item types, but not system types or predefined part types. The item type list in the system administration client is the list you can most easily control and adding to the list is the initial cause of the limitation.

However, the actual limitation is usually based on more item types than what appear in the system administration client list and is based on the count of that list. The actual maximum count and list expanded varies by query depending on the query constraints which varies by feature, configuration settings, and application. The actual limitation is usually based the total number of all item types in the system, including system types and predefined part types. Therefore, you must consider access to the all item types in the system and you are not
necessarily limited to only 19 item types in the system administration client list. When this error occurs, the log file includes a breakdown of all item types in the system for your review.

If you exceed the maximum number of item types supported for the operation, you get an error message CQL154 with CQL057 and the log file shows original server error state SQL RC = -129. You must do one or more of the following actions:

- Restrict access to item types for users so that they have access to a maximum of 19 item types. Or, reduce to the actual number of item types that enables the feature to work based on the feature, configuration settings, and application you are using.
- Restrict access to item types for predefined compound document part types that you do not use if you do not use other applications that use them. For example, ICMANNOTATION, ICMBASESTREAM, ICMNOTELOG, or any others you might not use in compound documents. The part types behave just like item types in queries and affect item type counts. Review the full list of part types listed with the error in the log file. You might need to edit and run Java native API samples to edit predefined non-system part types.
- Restrict access to predefined item types that you do not use if you do not use other applications that use them. For example, ICMSAVEDSEARCH, ICMFORMS, and ICMDFOLDERS. Review the full list of item types listed with the error in the log file.

Folder types:

IBM CMIS for Content Manager uses the designated folder types or default folder type ClbFolder, to create new folders. The ClbFolder item type is a CMIS-optimized item type.

In Content Manager EE, all items have a corresponding item type, including folders. Documents and folders are both items, and they have corresponding document types and folder types, which are both item types. All item types support metadata.

An item type defines the list of fields and properties the item supports. The item type also defines how the item supports binary content, for example, as an optimized single-part resource item, multi-part document model, or metadata only (no binary content).

Content Manager EE does not restrict folders by item type. Item types do not have constraints to limit use to only documents or only folders. Therefore, a folder is not determined by item type, but instead by the semantic type that is used when a new item is created, on an item-by-item basis. Any item type can be used for a folder or document. However, applications can separate document types from folder types by convention, which is recommended as a best practice. Many applications use only one item type for creating folders. However, all applications should tolerate folders returned from any item type and handle data that was created by other applications that might not use the same convention. Some Content Manager EE applications allow users to choose folder types.

You can still view, browse, and open existing folders of any item type that were created by other Content Manager EE applications.
Limitations when creating folders in the All Item Types library

The All Item Types library is a special, limited library that shows all items by their document type, not by folder hierarchy. The first level of children within this library are special virtual item type folders that represent each document type. These virtual document type folders show all items that use the particular item type, up to a configurable maximum number, including any documents or folders. However, the ClbFolder item type is hidden from the All Item Types library because this folder type is intended only for folders, not documents, and therefore does not match the purpose of sorting documents by document type.

You cannot save an item directly within an item type folder if its item type does not match the folder item type. The default document type that is used for documents (not folders) automatically uses the type indicated by the item type folder.

You cannot create a folder in a document type folder, because a document type folder can only contain document item types. For example, if you try to create a new folder directly in the CmisDocument item type folder in the All Item Types library, you will receive an error because you cannot create an item of type ClbFolder in a folder that only contains items of type CmisDocument.

You cannot organize the contents in an item type folder using subfolders. Document type folders are not intended to be used as a browse paradigm.

The contents of any subfolders that are directly within an item type folder are unrestricted. For example, if another application creates a folder using an item type called MyItemType, then the children in the folder .../ALLITEMTYPES_LIB/MyItemType/mySubFolder will not have any restrictions on their item types. The subfolder is not restricted to contain only items of type MyItemType. However, any documents that are created in subfolders will use MyItemType as the default item type if no type specified.

Parent topic: "Working with documents" on page 119

Related concepts:

"Document types optimized for CMIS"

In IBM CMIS for Content Manager, a document is known as a converted, extended, or CMIS-optimized document type if it has extended metadata for the standardized name attribute and databases indexes. CMIS-optimized document types support optional extended metadata that adds features and optimizes performance with IBM CMIS for Content Manager.

Document types optimized for CMIS:

In IBM CMIS for Content Manager, a document is known as a converted, extended, or CMIS-optimized document type if it has extended metadata for the standardized name attribute and databases indexes. CMIS-optimized document types support optional extended metadata that adds features and optimizes performance with IBM CMIS for Content Manager.

The IBM CMIS for Content Manager work best with CMIS-optimized document types. The optional extended metadata supports additional fields and properties in the CMIS data model and optimizes behavior for the documents. You can add different subsets of additional fields to the CMIS-optimized document types. The optional extended metadata is grouped into special attribute groups that are hidden from property sheets and are supported only within the attribute group. Although you must add the attribute group to document types, support for each
field is detected by existence of the individual attribute within the group, such as if an attribute is omitted in a document type or filtered out by an item type subset view. You can customize attribute and attribute group definitions for each document type, such as choosing a different maximum length value, default value, or other constraint.

CMIS-optimized document types support one or more of the following optional extended metadata groups:

**Universal name**
Name ICM$NAME (repository standardized name attribute)

**Content**
- Label, or filename (deprecated)
- Title
- Description
- Authors
- Categories
- Effective date
- Is hidden
- Language
- Owners

**Version**
- Version label
- Version comments

The fields use only the attribute within the correct group, not independent attributes outside of the group or attributes that belong to different groups. If they are added as independent attributes or as a different group name, the attributes function only through property sheets and are not associated with the field. The only exception is the repository standardized name attribute, ICM$NAME, which must exist as an individual attribute without an attribute group.

The most important optional extended metadata group is the Content group or the universal standardized name attribute. Specifically, the most important extension is the either the universal standardized name attribute, ICM$NAME, or the alternate standardized name attribute Label in the Content group. The existence of the Label attribute or standardized name attribute alone defines whether a document type detected as CMIS-optimized in terms of behavior. Without the standardized name attribute, documents and folders are unconverted document types, which are not locatable by name alone and are supported in a compatibility mode, as described in the next section.

**Important**: Do not add extended metadata to document types manually, especially the Content group with the standardized name attribute, Label. Extending metadata manually significantly degrades performance and scalability, which are not supported properly if improperly extended document types exist in the system. Also, adding certain extended metadata without also creating the correct database indexes can significantly degrade performance and scaling for all CMIS-optimized document types due to the existence of non-optimized but CMIS-enabled document types in the system.
Unconverted document types

IBM CMIS for Content Manager also supports document types that are not CMIS-optimized, and do not have the optional extended metadata. Therefore, you can work with your existing documents and document types, which are called unconverted document types. Unconverted document types often work just as well as converted document types in IBM CMIS for Content Manager. However, in some cases unconverted document types must operate in a compatibility mode that uses a best-effort mapping that might not be optimal. Also, unconverted document types do not support persistence of any field that requires extended metadata to persist the value and cannot be mapped using a best-effort mapping. However, if path indexing is enabled, all name and path-related behaviors become equally optimized as any other optimized document type or folder type even if the document type or folder type is not extended or optimized.

For example, the title and description fields are ignored and not persisted for unconverted document types. Also, the name might not always be mapped correctly when retrieving individual document properties unless path indexing is enabled.

Even unconverted document types support names (labels), in most cases, but the name mapping varies by document type, item type subset view, or sometimes by item. If available, the native Content Manager EE original file name field is mapped from the following:

- Documents from resource-classified document types, optimized single-part documents
- Document parts from document-classified document types, multi-part documents

However, the native field is not always available for multi-part documents if there are no parts or the parts are not accessible or not retrieved for performance reasons. If the native original file name field is not available or not retrieved, IBM CMIS for Content Manager detects the best possible attribute that represents a name on the root component of the item. Many custom document types have an attribute that contains a name, but is not standardized. If you have a name attribute, flag it as representative in the document type definition to indicate that it is your name for the item and can be used as a file name. You might see different names, depending the limited data that is retrieved to optimally perform a particular operation, and depending on the item type subset view you have access to. If path indexing is enabled, the fully mapped name is indexed and might be reused for better behavior.

Also, compared with CMIS-optimized document types, unconverted document types are more dependent on the application’s ability to recognize important state changes and use the fast path or ID that is reported by IBM CMIS for Content Manager. If a document changes states and the application does not automatically refresh its references or use the updated references returned by the services after a state change, you will need to manually refresh your folder view or close and reopen your document to continue working with it. Depending on the application, if you do not refresh your folder view or close and reopen your document, subsequent operations might report an error, report that the document has been deleted or is not found, or do nothing at all. In some cases, the application might save a new copy of the document instead of updating the existing document. Important state changes include:

- Creating a new document or draft
- Checking in an unpublished document draft for the first time
• Moving an item
• Saving a new copy of a document

Smaller state changes include:
• Saving changes as a draft
• Checking in any document

In some cases, the document transforms into an entirely different document, such as when checking in a document for the first time when it becomes a published document for the first time. Also, the fast path and ID for document objects are different than the fast path and ID for drafts of their parent documents.

Tip: Enable path indexing for optimized behavior for all state changes affected by path changes. Path indexing optimizes behavior for both unconverted and optimized item types.

Tip: IBM CMIS for Content Manager will cache the corrected mapping for a short, configurable period of time. For example, if you drag and drop a document in a library that defaults to an unconverted document type, you will be able to open the document during the short period of time that the new document path is temporarily cached by the services. If you wait longer than the specified period of time that the path is cached, nothing will happen when you try to open the document, or it will be reported as deleted or not found. The unrevised path mapping cache helps applications that create a document in multiple steps and reuse the proposed path for each step rather than the fast path or ID that was returned when the document was first created. The cache also helps users work with documents that they just created without having to refresh the folder view in Windows Explorer.

However, if you are using a CMIS-optimized document type, IBM CMIS for Content Manager can detect and correct the outdated IDs and fast paths so that you can continue to use the document without refreshing, even if the application does not recognize the change. CMIS-optimized document types use the configuration setting findDocumentWithoutFastAccessPath to find the document by name when the ID or fast path is outdated.

Example

You might need to refresh your folder view or close and reopen your document in the following situations without path indexing or optimized types:

• If you check in an unpublished document from an unconverted document type for the first time, you will not be able to access the document until you refresh your folder view.
• If you open an existing document in Microsoft Word and use Save As to save the document as a different copy, any subsequent saves or check-in operations will work only within the short time period when the new path is cached temporarily by IBM CMIS for Content Manager. After that time, IBM CMIS for Content Manager will save a separate draft copy of the document.
• If you move a document that is from an unconverted document type, you will not be able to access the document until you refresh your folder view.
• If you copy or create any new documents from unconverted document types, you will not be able to access the document until you refresh your folder view after the document path is no longer cached.
“Data model extensions”

The groups of IBM CMIS fields are organized into attribute groups (starting with clb). Some fields are supported by independent attributes without an attribute group.

Parent topic: "Working with documents” on page 119

Related concepts:

“Folder types” on page 128
IBM CMIS for Content Manager uses the designated folder types or default folder type ClbFolder. to create new folders. The ClbFolder item type is a CMIS-optimized item type.

“Model mapping” on page 135
IBM CMIS for Content Manager maps IBM CMIS object types to Content Manager EE item types.

Data model extensions:

The groups of IBM CMIS fields are organized into attribute groups (starting with clb). Some fields are supported by independent attributes without an attribute group.

The following groups of IBM CMIS fields are available as extensions to item types:

Independent
Includes the following field: label (name)

Content standard metadata
Includes the following fields: label (deprecated), title, description, authors, categories, effective date, is hidden, language, and owners.

Version info
Includes the following fields: version comments and version labels.

Application-controlled system attributes copy (deprecated)
Overrides the system control of the following fields: created, creator, modified, and modifier. You can also override the following binary resource fields: data length, data MIME type, and data modified.

Important: If CMIS fields are added, the applications control these fields, not the system. All applications must set and maintain these fields. The values are set, updated, and retrieved exactly as specified by the applications. If the values are not set by the applications, the fields are null. Adding this extension is not recommended because most applications depend on the system for setting and maintaining values for these fields. Adding this extension might negatively affect applications that depend on values that are not null values for these fields.

Some attributes in these groups must be marked with specific constraints, such as required minimum and maximum values or lengths, and default values for the best interoperability with all applications and with database indexes for performance and scaling. You can add the required attributes only to the new document types that do not have any data store using them. The DocumentTypeEnable tool checks whether the data exists. If no data exists, the appropriate attributes are marked as required. If the data exists, such as if you are extending an existing document type, then all attributes are not required. For existing items or items created by other applications without the required values, the items behave in a mixed compatibility model so that the values are mapped.
when required. However the values might not be always available, and the mapping can vary depending on operation. The primary attribute that is required is the label or the file name.

You can optionally update your existing items to enter the values. The following tables lists the field groups that are available as extensions for making document types optimized for CMIS.

### Table 44. Field Groups that are available as extensions

<table>
<thead>
<tr>
<th>Extension group</th>
<th>Definition name</th>
<th>Display name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>clbContent</td>
<td>Clb-Item Content</td>
</tr>
<tr>
<td>Version</td>
<td>clbVersion</td>
<td>Clb-Item Version</td>
</tr>
<tr>
<td>AppCtrlSysAttrs (not recommended)</td>
<td>clbAppCtrlSysAttrs</td>
<td>Application-Controlled System Attributes Copy</td>
</tr>
<tr>
<td>AppCtrlSysResAttrs (not recommended)</td>
<td>clbAppCtrlSysResAttrs</td>
<td>Application-Controlled System Resource Attributes Copy</td>
</tr>
</tbody>
</table>

### Table 45. Fields

<table>
<thead>
<tr>
<th>Group</th>
<th>Field names</th>
<th>Definition names</th>
<th>Data types</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>label</td>
<td>ICM$NAME</td>
<td>Varchar(255)</td>
<td>Required, minimum length 1</td>
</tr>
<tr>
<td>Content</td>
<td>Label</td>
<td>clbLabel</td>
<td>Varchar(255)</td>
<td>The following fields are required:</td>
</tr>
<tr>
<td></td>
<td>Title</td>
<td>clbTitle</td>
<td>Varchar(255)</td>
<td>• Label</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>clbDescription</td>
<td>Varchar(510)</td>
<td>• Title</td>
</tr>
<tr>
<td></td>
<td>Authors</td>
<td>clbAuthors</td>
<td>Varchar(400)</td>
<td>• Is Hidden</td>
</tr>
<tr>
<td></td>
<td>Categories</td>
<td>clbCategories</td>
<td>Varchar(255)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Effective Date</td>
<td>cmbEffectiveDate</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is Hidden</td>
<td>clbIsHidden</td>
<td>Short</td>
<td>The clbIsHidden field is a BOOLEAN field and must have a defined range of [0,1], with the minimum value as 0 and maximum value as 1.</td>
</tr>
<tr>
<td></td>
<td>Language</td>
<td>clbLanguage</td>
<td>Varchar(25)</td>
<td>Important: The label is the standardized name attribute. The label value is a file name or short name, such as CMISRelease.doc. The long name of the file is the file name only without the extension, such as CMIS Release.</td>
</tr>
<tr>
<td></td>
<td>Owners</td>
<td>clbOwners</td>
<td>Varchar(255)</td>
<td></td>
</tr>
</tbody>
</table>

| Version          | Comments     | clbVerComments   | Varchar(500) | |
| AppCtrlSysAttrs  | Version labels | clbVerLabels    | Varchar(255) | |
| AppCtrlSysResAttrs | Created     | clbCreatedTS     | Timestamp    | deprecated |
|                  | Creator      | clbCreatedBy     | Varchar(32)  | |
|                  | Modified     | clbUpdatedTS     | Timestamp    | |
|                  | Modifier     | clbUpdatedBy     | Varchar(32)  | |
|                  | Data length  | clbBinaryLength  | Long         | deprecated |
|                  | Data MIME type | clbMimeType      | Varchar(20)  | |
|                  | Data modified | clbBinaryLastModified | Timestamp | |

Parent topic: "Document types optimized for CMIS" on page 129
Model mapping:

IBM CMIS for Content Manager maps IBM CMIS object types to Content Manager EE item types.

Basic mapping: IBM CMIS for Content Manager and Content Manager EE

The following table shows the basic mapping between IBM CMIS for Content Manager and Content Manager EE:

<table>
<thead>
<tr>
<th>IBM CMIS</th>
<th>Content Manager EE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object type</td>
<td>Item type</td>
</tr>
<tr>
<td>Properties</td>
<td>Attributes</td>
</tr>
</tbody>
</table>

"Document type and folder type mapping"  
CMIS object types are the combined set of document types and folder types. CMIS object types are mapped directly to Content Manager EE item types.

"Property mapping" on page 136  
CMIS properties are mapped to Content Manager EE attributes.

"Property type mapping" on page 137  
CMIS property values are mapped to the appropriate target type in Content Manager EE by using best-effort mapping.

Parent topic: “Working with documents” on page 119

Related reference:

Permissions mapping

The applications enable the users to run various operations and each operation requires a special privilege or set of privileges. The users must have the privileges that are required to access Content Manager EE repository.

Document type and folder type mapping:

CMIS object types are the combined set of document types and folder types. CMIS object types are mapped directly to Content Manager EE item types.

Additionally, in Content Manager EE, administrators can optionally define item type subsets, or views, which filter the metadata fields and properties to a subset of the item type. The item type subsets are intended to be the only subset accessible to a set of users. An administrator usually restricts access to the full base item type view, which is the automatic default view with the same name as the item type, and grants different users access to only one item type subset view. Then, users with different levels of access see different item type subsets of the same base item type. Item type subset views are supported and enforced by IBM CMIS for Content Manager.

Although the item type subset views are named independently from the item type name, with the exception of the automatically-created default base item type view, the item type name, the document type, and the folder type name remains the same. Even though the document type and folder type name is the same, users see the document type and folder type properties only that are accessible through their active item type subset view. The active view for the user is the first view the user has access to. The active view is displayed to the user at run time for all operations. Content Manager EE always uses views, even if it is the
automatically-created default base view. The default base view shares the same name as the item type and exposes the full item type metadata.

The CMIS ID is stored in the CMIS property, cmis:objectTypeId. The document type and folder type name and ID report the name of the base item type and are not based on the current item type subset view that a specific user has access to. The subset view that is in effect is determined at run time based on the current user. Therefore, users and applications can share document type and folder type names and IDs between users who can collaborate on the same documents and document type (or folders and folder types). The correct item type subset is automatically applied based on each user's access. Each user sees the item through the appropriate subset. The users can also share the same link or ID without specializing links or IDs to be specific to a particular subset view. For example, a manager with access through one item type subset can send a document link to an employee who sees a different item type subset and the employee can open the document.

When users list document types and folder types, they see all the document types and folder types for which they have access to, at least one item type subset view including the base item type. However, the document type and folder type definitions will show the properties only that are accessible in a user's active subset view. Properties are listed only for those that are accessible with the current user's active subset view. For example, if two users look at the same document, folder, or object-type definition, one user might see more properties than the other user, if one has a more restricted view.

Tip: The item type classification resource is the recommended classification to use with IBM CMIS for Content Manager. The resource item type classification represents optimized single-part documents. Item type definitions with the document item type classification, which represent multiple-part documents, are supported but they do not perform as well as resource item type documents since they are not optimized for a single binary content. Document-classified item type documents operate in a compatibility mode and can experience some behavior limitations. Item type definitions with the classification item or standard, which represent no-content documents, are supported as zero-byte documents with full metadata support and cannot be created or updated with more than zero-length binary content.

Parent topic: Model mapping on page 135

Related concepts:

- Item types on page 123
- Item type classifications on page 124

The server can contain many item types, including item types that are used by other applications that interoperate on the same repository. You must ensure that you configure your system so that IBM CMIS for Content Manager users select the appropriate item type when adding documents in.

For best performance and behavior optimized for normal documents, use item types with item type classification resource. Item types with the classification of resource are optimized for a single binary resource and work best for normal documents.

Property mapping:

CMIS properties are mapped to Content Manager EE attributes.
Content Manager EE attributes sometimes belong to attribute groups. The CMIS property ID, cmis:id is the qualified attribute name in Content Manager EE, which uses the following syntax when using attributes that belong to an attribute group:

```
AttributeGroupName.AttributeName
```

If the Content Manager EE attribute does not belong to an attribute group, a special attribute group ID of clbNonGroup is used instead. For example, the property ID of the SOURCE attribute in the Content Manager EE item type NOINDEX is:

```
clbNonGroup.SOURCE
```

The local name tag (cmis:localName) of the property maps to the qualified attribute name in Content Manager EE. The display name tag (cmis:displayName) maps to the translatable description of the attribute definition, which varies depending on the IBM CMIS for Content Manager system locale. Content Manager EE administrators can add or change translated description for multiple locales at any time in the Content Manager EE system administration tool.

Properties are also limited by other Content Manager EE attribute constraints, such as required, minimum or maximum lengths and sizes, unique value, default value, or other constraints.

**Important:** Unique constraint means that the property value must be unique among all documents with the same document type, not only within the parent folder, and not compared against documents from other document types even in the same parent folder.

You can only use the properties that are defined for the document type, based on the subset that the user has access to. Users with access to different item type subsets might see different properties, as described in the previous sections. IBM CMIS for Content Manager does not accept properties that are not defined for the document type or are not accessible in the user's active item type subset view. If properties are submitted that are not defined in the document type definition as listed through the current user's subset view, an error is reported.

You can submit only one value for a property. IBM CMIS for Content Manager does not accept multiple values. If the value array for a property contains more than one element, an error is reported.

A zero-length array value is treated as a null value. If a property is submitted with null value or an empty array, the property will be set as null, if the corresponding attribute is defined as nullable. If the attribute is not defined as nullable, an error is reported. Depending on the application, if you do not want to set one or more of the properties within an object type, submit only the properties that you want to set, and no changes from the persisted value will be made to the properties that you omitted. Any properties that are restricted by the current item type subset view are treated as omitted.

**Parent topic:** [“Model mapping” on page 135](#)

**Property type mapping:**

CMIS property values are mapped to the appropriate target type in Content Manager EE by using best-effort mapping.

Use the CMIS property types and appropriate Content Manager EE attribute types as recommended in the tables in the next section.
The property definitions include a property type `cmis:propertyType`, which identify the property type that must be used for creating folders or documents and for updating folders and documents. When retrieving folders and documents, the values will always be returned by using the same property type.

The tables in the next section show which property types are used for specific Content Manager EE attribute types.

Refer to this table to determine what CMIS property types are for a particular Content Manager EE attribute type or any time you need a quick reference.

**Property type mapping reference**

Refer to this table to determine what CMIS property types are for a particular Content Manager EE attribute type or any time you need a quick reference.

**Property type mapping: Content Manager EE to CMIS**

The following table shows how Content Manager EE attribute types map to CMIS property types:

```
<table>
<thead>
<tr>
<th>Content Manager EE attribute types</th>
<th>CMIS property types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB (binary large object)</td>
<td>Omitted</td>
</tr>
<tr>
<td>Char (fixed-length character string)</td>
<td>String</td>
</tr>
<tr>
<td>CLOB (character large object)</td>
<td>String</td>
</tr>
<tr>
<td>Date</td>
<td>Datetime (date portion only)</td>
</tr>
<tr>
<td>Decimal</td>
<td>Decimal</td>
</tr>
<tr>
<td>Double</td>
<td>Decimal</td>
</tr>
<tr>
<td>Long integer</td>
<td>Integer</td>
</tr>
<tr>
<td>Short integerInteger</td>
<td>Integer</td>
</tr>
<tr>
<td>Varchar (variable-length string)</td>
<td>String</td>
</tr>
<tr>
<td>Time</td>
<td>Datetime (time portion only)</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Datetime</td>
</tr>
<tr>
<td>Reference</td>
<td>Omitted</td>
</tr>
</tbody>
</table>
```
Attention:

- The CMIS property type strings must be in lower case.
- Property values that have the String property type can usually be converted to a Content Manager EE attribute type, but not always. If the submitted type and target type are incompatible, an error is reported.
- Date and Time attributes might not reflect the original intended moment in time when viewed by different time zones and geographies. The Timestamp attribute is strongly recommended instead of separate Date and Time attributes. For more information, see the tips that are documented in the next section.

Property type mapping: CMIS to Content Manager EE

If you already have a CMIS type, you want to find what Content Manager EE attributes will map to it, use the following recommended mappings:

Table 48. Property mapping: CMIS to Content Manager EE

<table>
<thead>
<tr>
<th>CMIS type</th>
<th>Content Manager EE type</th>
<th>Recommended Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datetime</td>
<td>Timestamp, Date, Time</td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Short integer or long integer</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>Varchar (variable-length) or Char (fixed-length)</td>
<td>Choose a reasonable maximum.</td>
</tr>
<tr>
<td>Decimal</td>
<td>Decimal or double</td>
<td></td>
</tr>
</tbody>
</table>

Consider the following information when you define attributes:

- If string values for a particular property are always the same length, use Char, which is a fixed-length string attribute, instead of Varchar, which is a variable-length string attribute. Char performs much faster than Varchar. However, Char automatically pads white space for any string submitted that is smaller than the specified length. This makes fixed length strings harder to query by the exact string if any values are smaller than the full size. For the most flexibility, Varchar is a good choice if you have strings that greatly vary in length.

- Choose reasonable maximum lengths for your string attribute definitions, because an item type's metadata is limited to 32K in the database. The actual metadata size limit depends on the tablespace size that is used when creating the repository database. Unicode strings can consume multiple bytes per character and can vary depending on actual characters in each string. Drafts of items with metadata that is less than approximately 25K perform much faster than drafts with more metadata. Draft instances with more metadata require expensive overflow storage that does not perform fast or scale for large numbers of such drafts.

- Always choose Timestamp over Date-only or Time-only attribute types if possible. For globalized applications, all dates and times are relative to the time zones and local daylight-savings times, which can vary depending on the day, year, timezone, and country. Only the Timestamp attribute preserves the exact moment in time. Any date or time is calculated based on a moment in time by applications and servers, but when only the date portion or only the time portion is persisted, the precision is lost and cannot be mapped back to the exact moment in time. Therefore, when viewed by applications in different time zones, the value might not reflect the original moment in time from the perspective of different time zones. Date-only fields generally report the original date in GMT. Time-only fields might not recognize daylight-savings time as it applies to the
exact moment in time, especially when viewed from different time zones. To preserve the local date without any time zone alterations, you can use a fixed-length string field.

- If you define item types in Content Manager EE with required BLOB attribute and then create a CMIS document with those Content Manager EE item types, you might get an error message saying that the required property does not contain a value. The item types with required BLOB attributes are for custom applications only and the custom applications can set the BLOB attributes. Because there is no MIME type associated with a BLOB, any application cannot set the BLOB attributes. This means that you cannot create new documents with required BLOB attributes. You can edit existing documents from this document type, but cannot view or edit the BLOB attribute. The BLOB attributes are not recommended as a best practice. The existence of document types with BLOB attributes degrades the performance and scalability because the document types with BLOB attributes are not optimized for multi-item data retrieval.

Parent topic: “Property type mapping” on page 137

Permission mappings in ACL services:

ACL services define a set of operations to retrieve and modify an object's Access Control List (ACL). ACL services are defined in the OASIS Content Management Interoperability Services (CMIS) specification.

CMIS ACL and Permission Definitions

By CMIS specification, an ACL is a list of Access Control Entries (ACEs). For more information, see http://docs.oasis-open.org/cmis/CMIS/v1.0/os/cmis-spec-v1.0.html#_Toc243905406

An ACE contains the following components:

- One Principal: A principal represents a user management object, for example, a user, group, or role. A principal contains one String with the principalid element.
- One or more Strings with the names of the permissions.
- A Boolean flag direct, where TRUE indicates that the ACE is directly assigned to the object. If FALSE, that the ACE is somehow derived.

The CMIS specification defines three basic permissions:

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmis:read</td>
<td>Permissions to read the content.</td>
</tr>
<tr>
<td>cmis:write</td>
<td>Permissions of cmis:read and the permissions to edit the content.</td>
</tr>
<tr>
<td>cmis:all</td>
<td>This permission is used to express all of the permissions of a repository and includes cmis:read and cmis:write.</td>
</tr>
</tbody>
</table>

The specification indicates that repositories can extend this set with repository-specific permissions.

IBM Content Manager privilege set and permission mapping

IBM Content Manager Access Control List (ACL) is a list of users or user groups, each with one privilege set. A privilege set is a collection of privileges, while a privilege is the right to access a specific object in a specific way. To describe this in another way, a user or user group is given access to a particular object by
associating the user or user group with a privilege set in the object’s ACL. The user or user group with a privilege set is called an Access Control Entry (ACE).

Mapping IBM Content Manager ACL to CMIS ACL gives the following special characteristics to IBM CMIS for Content Manager ACL services:

- In IBM Content Manager, privileges are assigned to a user or user group via privilege set. An ACE can have only one privilege set for the user or user group. When mapping to a CMIS ACE, the CMIS ACE can have only one permission.
- Because the CMIS permission is mapped to a IBM Content Manager privilege set, the privilege set must be predefined in IBM Content Manager before it can be assigned to an user or user group to form an ACE.

To convey the native security settings of IBM Content Manager objects to CMIS compatible client applications without losing information and while supporting the CMIS specification better, IBM CMIS for Content Manager uses a mixed approach to map between a IBM Content Manager object’s privilege sets and a CMIS object’s permissions; the permission string in a CMIS ACE can be either CMIS basic permission or IBM Content Manager privilege set name.

The permission mapping occurs in both directions between IBM CMIS for Content Manager and CMIS compatible client applications. The mapping is controlled by a set of configuration parameters in IBM CMIS for Content Manager configuration file `cpathservice.properties`. The parameters define what IBM Content Manager privilege set is mapped to a CMIS basic permission, for example, considering the following mapping:

```
cmisToCMPermissionMapp.Read = ClientUserReadOnly
```

This mapping means that `cmis:read` corresponds to the IBM Content Manager privilege set `ClientUserReadOnly`.

**Downstream permission mapping**

The permission mapping from a IBM Content Manager privilege set to permissions received by CMIS compatible client applications is computed by the algorithm below.

1. An IBM Content Manager privilege set is attempted to be mapped to a CMIS basic permission by the mapping definitions in the `cpathservice.properties` configuration file. If the mapping is found, the CMIS basic permission is set to the permission of the principal.
2. Following step 1, if the IBM Content Manager privilege set cannot be mapped to any CMIS basic permission, the privilege set is set to be the permission of the principal. The string representation of the permission is the privilege set name with the following prefix: `cm`.

For example, if the `cpathservice.properties` file has a mapping definition, `cmisToCMPermissionMapp.Read = ClientUserReadOnly`, and the principal’s privilege set is `AccountantPrivSet`, the permission mapping result will be `cm:AccountantPrivset` in the response to the CMIS compatible client applications. However, if the principal’s privilege set is `ClientUserReadOnly`, the permission mapping result will be `cmis:read`.

**Upstream permission mapping**

The permission mapping from CMIS compatible clients to IBM Content Manager privilege set is computed by using the following algorithm:
1. Only CMIS basic permissions (cmis:all, cmis:write, and cmis:read) and IBM Content Manager privilege set names with a cm prefix are allowed.

2. CMIS basic permissions are mapped to IBM Content Manager privilege sets according to the mapping definitions in the cmpathservice.properties file.

3. Prefixes are moved from privilege set name with the cm prefix and they are treated as they are: as a privilege set.

For example, assume that the cmpathservice.properties mapping file has the following mapping definition:

```
CmisToP8PermissionMapp.Document.Read = ClientUserReadOnly
```

If the client requests to modify an object's permission for a principal to cmis:read, then the principal's privilege set to the object in IBM Content Manager will be changed to ClientUserReadOnly. If the client requests to modify the object's permission for the principal to cm:AccountantPrivset, the principal's privilege set will be AccountantPrivset.

If the client requests to modify the object's permission for the principal to cmis:read, cm:AccountantPrivset, an error will be returned to the client because only one permission is allowed in the ACE in the request.

Parent topic: “Working with documents” on page 119

Paging:

Paging enables you to view subsets of results, one page at a time, instead of retrieving and scrolling through the full result set. A page of results contains a subset of items from a potentially larger set of results.

To browse most the results from beginning to end, retrieve all results in one call on a large single page of results instead of paging. However, to browse a small subset of results, retrieve only a small set of results and request more as you need them. Each page request runs the same query as a non paged request or single large page request. Although running the query involves a performance cost that is constant whether you are using paging or not, you save the cost of data retrieval, transmitting, and rendering results for data on other pages and retrieve only the data for the page you are viewing. Retrieving the first page of results is much faster than retrieving the full set of results. The performance cost of retrieving each subsequent next page is progressively more than the previous page because of some overhead to process and skip past the previous page of results. Retrieving the last page of results is much more expensive than retrieving the first page, but the last page is still less expensive than retrieving the full set of results. However, browsing through all pages of results is more expensive than retrieving the full results instead of paging through them. Browsing through only a few pages of results can be more or less expensive than retrieving the full results depending on the query complexity, number of results, data size, transmission speeds, and rendering speeds. For example, the combined cost of a few pages can cost more than retrieving the full results. However with pages you do not have to wait as long for each page.

To minimize the number of pages you must browse through, sort the request so that the results you want are on the first page. For example, it is common to sort by last modified date in descending order to see the most recently active documents that you look for. You can also use a query that returns a smaller subset of results that you want on the first page. For example if you are running a saved search for documents you modified, use a saved search instead that finds documents that you modified last in the past 24 hours. Because the results are
sorted by the database before paged, you can only sort on results that are
guaranteed to be on all possible results for that query in the database. The sorting
support varies by feature, by configuration settings, and by data. For example you
can sort libraries by file name, but you cannot always sort saved searches by file
name.

Some applications request paged results one page at a time, such as the custom
library. Some applications allow you to change the page size so that you can select
how large each page of results is. If you can change the page size, you can make a
non paged request by requesting a large single page of results and get all results
on the first page. If the application allows you to configure the default page size,
you can select a page size that balances performance of first page rendering, user
experience for the size of the page, and minimize the number of multiple page
requests that are required.

IBM CMIS for Content Manager supports paging of the following result lists:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported property sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folder or library children</td>
<td>• Name&lt;br&gt;• Created date&lt;br&gt;• Last modified date&lt;br&gt;• Limited support for label and title by special folders only, such as, the All Item Types library and some document type folders.</td>
</tr>
<tr>
<td>Document types or folder types list</td>
<td>• Name&lt;br&gt;• Custom user properties that exist in all the results. CMIS properties including creation date, last modification date. Limited support for name and content stream length, and text search score.</td>
</tr>
<tr>
<td>Search results</td>
<td>• Custom user properties that exist in all the results. CMIS properties including creation date, last modification date. Limited support for name and content stream length, and text search score.</td>
</tr>
</tbody>
</table>

"Paging limitations"
Consider the following limitations before you start using the paging capability of IBM CMIS for Content Manager.

Parent topic: "Working with documents" on page 119

Paging limitations:
Consider the following limitations before you start using the paging capability of IBM CMIS for Content Manager.

Limitations
The limitations of paging in IBM CMIS for Content Manager are:

Drafts sorted at location of the published parent document
By default, some applications that display results by the page and that display drafts, if they are more current than their published equivalents, sort the results based on the value of the published document. This means that the sort value of the published document determines the location of the draft document in the results.

These applications also display the date that the draft was created or last modified, not the date that the published document was created or last checked in. For example, suppose you created a published document on 09
December 2009, checked out that document on 15 January 2010, then created a new draft by saving changes on 20 January 2010. If you viewed results that contained this document, sorted by last modified date, the draft version of the document would be displayed with a creation date of 20 January 2010 and a last modified date of 20 January 2010. But the draft would appear in the result set where the published document would otherwise be: 09 December 2009, the last modified date of the published document. (Note that checking out a document does not modify it. A document is modified only if changes are detected during checkin.)

You can work around this limitation by using the following configuration parameters:

- `sortChildrenDraftValuesInline`
- `sortSearchDraftValuesInline`

**Result not valid entry in results**

When you query only for documents or when you open a saved search folder, and when results are paged, the results might include folder result and draft result placeholders for results that are not valid when the query expression is not written precisely. When paging, some results are replaced by placeholders and cannot be filtered out because of errors by the administrator constructing the query expression. The query expression returns results that should not be displayed to the current user or restricted in a particular feature. For example, if you define a saved search to find drafts and do not restrict it to return only drafts for the current user, you will see draft result placeholders in the results. The saved search for drafts must be defined by using the current user variable. The result placeholders that are not valid provide information about what is wrong with the non optimal query without breaking the entire query because of a single invalid result among many valid results. Sometimes, the administrators might find some valid queries that can include a few folder result placeholders for libraries and library groups or applications that match the query expression. The libraries and library groups or applications are document objects in the repository. Therefore, if an administrator creates a library and then runs a Lotus view or saved search to find all documents they created, the library is included in the result but must be replaced by a placeholder when returned by an interface that supports only document results. You might notice the folder results when prototyping or as an administrator, but these might be few and typically not seen by users.

**Continuity between page requests**

The page results can change between page requests and subsequent page requests reflect the current content of those pages as evaluated at the time each page request is run. The page results on the previous or next page are not guaranteed to be sequential continuing from or leading to the prior page results. By the time you open the next page, the total results might have changed with added objects, deleted objects, moved objects, or changed permissions. Each page reflects the current results in the page series as defined by the page size according to the current total result set. So the next page can return some objects already seen on the prior page or skip some results between the two pages if the result set has changed between page requests. For example, suppose 26 alphabetic letters are sorted alphabetically A-Z and paged with page size 5. Page 1 contains A, B, C, D, E, page 2 contains F, G, H, I, J, and page 3 contains K, L, M, N, O. But if you view page 1 (A-E), and while viewing page 1, two new letters are added A1 & B1, then when the user changes from old page 1 (A-E) that
is still showing to page 2, they see the next page D, E, F, G, H repeating D & E they saw on the last page. The series has changed and page 1 is now defined as A, A1, B, B1, C. Similarly, if you were viewing original page 2 (F-J) and went back to previous page 1, you see A-C and skip over D-E which became part of page 2 while they had the old page 2 still open. The page numbers are always evaluated according to the current result set.

There is no guaranteed continuity between page requests or meaning of page numbers between page requests. Each are evaluated independently.

However, if the result set does not change between page requests, the results are continuous with no break between pages. You can page through the entire result set without skipping or overlapping any results between pages if the full results that you are paging do not change. This limitation applies only when the result set changes while you are paging, such as results added or deleted, permissions granted or revoked, database reorganizes, or you change the sort order or page size.

**Sorting folders before documents**

By default, IBM CMIS for Content Manager does not support sorting folders before sorting documents. However, results might still be sorted within the page of results, but varies by feature and application. Sorting is limited within each page only and not across multiple pages. For example, you have 21 items in a folder, two folders, and 19 documents and your page size is 20. After sorting, the page 1 displays one folder named Zebra followed by 19 documents sorted alphabetically within page 1 starting with apple.txt to zoo.txt and on page 2 you could have one folder named Banana. Therefore, page 1 is sorted within its page, but not sorted across the two pages.

You can work around this limitation by using the following configuration parameters:

- `sortChildrenFoldersBeforeDocuments`
- `sortSearchFoldersBeforeDocuments`

**Default sort order defined by query expression**

When you query only for documents or when you open a saved search folder, the default sort order is determined by the Content Manager EE query expression constructed by the administrator. An administrator can change the sort order by modifying the Content Manager EE query expression to include the appropriate "SORTBY" syntax at the end of the query expression. The user or application can change the sort order on saved search results to sort instead on a limited number of Content Manager EE system attributes, such as created date and last modified date, ascending or descending, but no optional extended metadata attributes, such as label (file name) or title.

**Reserved keyword SORTBY if no default sort order defined**

When you query only for documents or when you open a saved search folder, the keyword SORTBY is reserved and cannot appear in anywhere in the query expression conditions or syntax except to declare the sort order in the query expression, unless the Content Manager EE query expression also includes a SORTBY order also at the end. The SORTBY keyword is reserved for uppercase, lower-case, and mixed-case characters. To support changing the sort order on an existing saved search, the query expression is cut at the last occurrence of SORTBY in the query expression and replaced with a new sort order. For example, you cannot run a saved search or parametric (metadata-based) search for documents with title, such as %SORTBY% unless there is also a SORTBY order at the end of the
query expression. You must define your query expressions with a default sort order or do not use the reserved keyword SORTBY anywhere else in the query expression. The conditions, such as document types, property names, property value constraints, and so on, are restricted only, not the values in the results returned. You also cannot use the reserved keyword within the SORTBY clause other than for the syntax to declare the clause. For example, you cannot sort on a property with the name abcSortBy.

**Sorting by Content Manager EE user name**

Most applications do not enable sorting on paged requests by user name properties, such as created by user and last modified user on paged requests. Sorting by user names in the database can only sort by the short name or login name for the repository that is stored in the database. However, the applications typically display the users display names obtained by an authentication server and the display names do not necessarily match the short user name for the repository.

**All Item Types library appears at end**

When the library listing is paged or sorted, the optional All Item Types library is always displayed on the last page of results regardless of sort order. The All Item Types library can be displayed at the end of the result set or sorted within the last page.

**Compatibility with sorting limitations**

Some applications can detect which properties are supported for sorting and limit sorting selection to only the supported properties. Other applications let you select all properties and do not support limiting sorting choices to only supported properties. If the application lets you sort by a property that is not supported on a paged result, you get an error message CQL188. Many applications enable you to submit a default sort order that might or might not be configurable. If the application supports detecting supported properties for sorting and an invalid default is selected, you must configure the application to use a valid default sort order and remember the setting for the next time you use the feature. If multiple sort orders are submitted, only the first valid sort order is recognized and any additional sort orders are ignored. This allows for applications that attempt multiple sort orders frequently to continue working if the primary sort order is supported. Only valid properties are allowed for a sort request. Any invalid property or property by which you cannot sort returns an error.

Parent topic: [Paging](#) on page 142

**Deleting folders:**

When you delete a folder, the deletion typically cascades to the folder contents too. All subfolders and documents that are visible to you are deleted along with the folder.

Any items that you do not have permission to see are hidden from deletion. The folder and all items that are visible to you will be deleted, but any items that are not visible to you will not be deleted. If the hidden items were filed only in the folder you deleted, the items become *unfiled*, which means they are not in a folder and they are accessible only through searching or by browsing in the All ItemTypes library.

When you delete a folder, the folder contents will be deleted even if they were *multi-filed*, which means they were filed in multiple folders. The items are deleted and removed from all multi-filed locations. For IBM CMIS, the `deleteSingleFiled`
option deletes all non-folder objects that can be filed whose only parent-folders are in the current folder tree. The option will unfile all other document or non-folder objects that can be filed from the current folder tree. Use the folderTreeDeleteOptionSingleFiledOnly configuration setting to mitigate the limitations of the cascade delete feature of Content Manager EE.

Most applications request cascade deletion by default. Depending on the service interfaces that are used by the applications, some applications specify an option to indicate cascade deletion, whether presented to the user or assumed as application behavior.

Cascading folder delete fails and rolls back deletion in the following situations:
• You do not have delete permission on the folder or any visible item within the folder.
• An item is locked by another user, and you do not have authority to override the lock.
• Hidden data model constraints prohibit cascade deletion, for example, reference attributes or foreign keys that depend on a delete rule.
• The folder tree is too large to delete in a single request.
• Any repository error occurs at any point during the cascade deletion.

Cascade deletion will complete, but will leave existing items as unfiled items in the following situation:
• You do not have permission to see the item or folder.

Parent topic: Working with documents on page 119

Versioning:

Versioning is the ability to store and maintain multiple versions of an item. You specify versioning rules when you define an item type. If an item type is enabled for versioning, all items in that item type can be versioned manually or automatically depending on the version control settings and the application support for versioning.

Versioning is handled by the Content Manager EE library server. Versioning capabilities are presented to the user through the client application user interface.

Parent topic: Working with documents on page 119

Version control settings:

For IBM CMIS for Content Manager, the version control settings of Content Manager EE affects the version control settings of IBM CMIS for Content Manager when you check in a document.

Version control settings:

For IBM CMIS for Content Manager, the version control settings of Content Manager EE affects the version control settings of IBM CMIS for Content Manager when you check in a document.

In Content Manager EE, document types are defined with one of three possible version control settings. The name of the setting can vary depending on the application that you administer or view the document type through. Depending on the interface, each of the three settings might be named differently, as follows:
1. Never or No Versioning
2. Manual, Application Controlled, Prompt, or Explicit
3. Always, Automatic, or Implicit

When Never create versioning is selected, IBM CMIS for Content Manager does not create new versions when a document is checked in or updated. There is always one version of a document.

When Always create versioning is selected, IBM CMIS for Content Manager creates a new version of the document whenever there is an update. You can create as many new versions as you want and you do not get an error message if you create several versions with the same content.

When Prompt to create versioning is selected, IBM CMIS for Content Manager creates a new version of the document if it is checked in as a major version. If the document is checked in as a minor version, the existing version is updated. For example, if you are working on document version 2, and call a checkin request with the major parameter set to true, a new version 3 is created. If you had set the major parameter to false, version 2 is updated. For updateProperties requests, the existing version is updated.

**Important:** As best practice, all operations must be performed from the private working copy (PWC) or draft, which can be created during check out of a document. User can perform all operations, such as set content and update properties on the private working copy that was created. User can check in or cancel check out on the private working copy. Based on version policy, the checkin operation might create a version or update the existing version of document. The cancel checkout operation on document deletes the private working copy.

The Version history and the properties of a document version display a snapshot of the document when the version was created or last modified. The version entries are copies of the document itself and have no separate version entry properties than the properties and binary content of the document. All values that are displayed pertain to the actual values of the document for the particular copy as a document version. Therefore, the created date and the last modified date are the dates when the document was created or last modified for a particular version. The creation date is always the date when the first published version of the document was created. Regardless of which version, the document was always created at the same original date. When you create the later versions of the document, the creation date is not updated and still displays the date when the document was first created. However, the last modified date shows the date when the document was last modified within the particular document version. The last modified date listed for the document version is usually the date when the version was created or last modified if the document version was modified after it was created.

**Parent topic:** "Versioning" on page 147

---

**IBM CMIS for Content Manager OnDemand implementation of the OASIS CMIS specification**

IBM CMIS for Content Manager OnDemand includes a limited set of read-only functions from the CMIS specification. IBM CMIS for Content Manager OnDemand does not support creating, updating, or deleting tasks.

IBM CMIS for Content Manager OnDemand Version 2.0.3, implements OASIS CMIS V1.0 with errata specification, which is available at: [https://www.oasis-](https://www.oasis-)
The implementation is based on IBM Content Manager OnDemand Web Enablement Kit (ODWEK) Java API.

One deployed instance of IBM CMIS for Content Manager OnDemand can be configured to work with only one IBM Content Manager OnDemand server.

**CMIS repositories** on page 150
A CMIS repository is mapped to a single IBM Content Manager OnDemand folder. Because searches in IBM Content Manager OnDemand are performed on folders and are limited to a single folder. Therefore, an IBM Content Manager OnDemand folder fulfills the OASIS Content Management Interoperability Services query specification requirement that requires that queries are scoped to a single repository.

**CMIS navigation services and root folders** on page 150
IBM Content Manager OnDemand does not have a root folder. Therefore, it implements the minimum root folder requirements of the OASIS Content Management Interoperability Services specification.

**Document retrieval** on page 150
IBM Content Manager OnDemand does not support OASIS Content Management Interoperability Services folders. Therefore, you must retrieve documents by using the document IDs, by running a query, or by using the unfiled service collection.

**Filed and unfiled documents** on page 151
All documents in IBM Content Manager OnDemand are unfiled. Documents in IBM Content Manager OnDemand are not filed in folders.

**Relationships** on page 151
Relationships are not implemented in IBM CMIS for Content Manager OnDemand.

**Policies and security** on page 151
Policies and ACLs are not implemented in this version of the IBM CMIS for Content Manager OnDemand.

**CMIS document and folder types** on page 151
Type information is retrieved from the IBM Content Manager OnDemand folder. An IBM Content Manager OnDemand folder is the mechanism through which users search for documents that have common properties; therefore, the folder contains the metadata information as IBM Content Manager OnDemand criteria.

**Documents** on page 151
An IBM Content Manager OnDemand document contains metadata and non-formatted data.

**Content streams** on page 152
The nonformatted data from an IBM Content Manager OnDemand document is retrieved in a byte array from the first content stream in the CMIS document.

**Versioning** on page 152
IBM Content Manager OnDemand does not support document versions, so the cmis:versionable attribute will always return false. The getAllVersions method will always return a single version. The checkout and checkin methods will issue notSupported exceptions or return empty lists if the method cannot issue this exception.

**Folders** on page 152
IBM Content Manager OnDemand does not have folders (OASIS CMIS folders that are used for filing documents). Therefore, IBM CMIS for Content Manager OnDemand implements the minimum folder requirements of the OASIS Content Management Interoperability Services specification.
The IBM CMIS for Content Manager OnDemand implementation for paging a list of query results is not efficient. Therefore, you should use queries that do not return a large number of results. If a query contains a large number of results, the performance of the IBM CMIS for Content Manager OnDemand server can be impacted.

IBM Content Manager OnDemand does not expose full text searching through the ODWEK API. Therefore the cmis:fulltextIndexed property returns a value of false.

IBM Content Manager does not support all of the query operations that are supported by OASIS Content Management Interoperability Services.

CMIS repositories

A CMIS repository is mapped to a single IBM Content Manager OnDemand folder because searches in IBM Content Manager OnDemand are performed on folders and are limited to a single folder. Therefore, an IBM Content Manager OnDemand folder fulfills the OASIS Content Management Interoperability Services query specification requirement that requires that queries are scoped to a single repository.

An IBM Content Manager OnDemand folder is the mechanism through which users search for documents that have common properties (IBM Content Manager OnDemand fields).

CMIS navigation services and root folders

IBM Content Manager OnDemand does not have a root folder. Therefore, it implements the minimum root folder requirements of the OASIS Content Management Interoperability Services specification.

The allowable actions that client applications can perform on the root folder are limited because IBM Content Manager OnDemand does not have a root folder structure. In addition, the root folder does not contain any child documents or folders.

Document retrieval

IBM Content Manager OnDemand does not support OASIS Content Management Interoperability Services folders. Therefore, you must retrieve documents by using the document IDs, by running a query, or by using the unfiled service collection.

You cannot use the following methods to retrieve documents because they return no values or cause exceptions:

- The getObjectByPath method returns a null value.
- The getFolder method returns a null value.
- The getFolderTree method throws a notSupported exception.
Filed and unfiled documents

All documents in IBM Content Manager OnDemand are unfiled. Documents in IBM Content Manager OnDemand are not filed in folders.

The `getUnfiled` method returns all of the documents in the IBM Content Manager OnDemand repository. All documents are returned when the `getUnfiled` is used because IBM Content Manager OnDemand documents are not filed in folders.

IBM Content Manager OnDemand does not have folders. Therefore, the `cmis:fileable` property returns a value of false.

Relationships

Relationships are not implemented in IBM CMIS for Content Manager OnDemand.

Policies and security

Policies and ACLs are not implemented in this version of the IBM CMIS for Content Manager OnDemand.

CMIS document and folder types

Type information is retrieved from the IBM Content Manager OnDemand folder. An IBM Content Manager OnDemand folder is the mechanism through which users search for documents that have common properties; therefore, the folder contains the metadata information as IBM Content Manager OnDemand criteria.

Because IBM Content Manager OnDemand does not have hierarchical folders, only the `cmis:document` and `cmis:folder` base types are available. The `CMISType.getSubTypes` method always returns an empty list.

IBM Content Manager OnDemand does not have folders. Therefore, IBM CMIS for Content Manager OnDemand implements the minimum `CMISTypeFolder` requirements of the OASIS Content Management Interoperability Services specification.

Because of the limited `CMISTypeFolder` implementation, the following properties return a value of false:

- `cmis:creatable`
- `cmis:fileable`
- `cmis:queryable`

Documents

An IBM Content Manager OnDemand document contains metadata and non-formatted data.
The allowable actions that client applications can perform on a document are limited because IBM CMIS for Content Manager OnDemand provides a read-only subset of CMIS functions. If a client application invokes a method to delete or update a document, IBM CMIS for Content Manager OnDemand throws a notSupported exception.

Parent topic: IBM CMIS for Content Manager OnDemand implementation of the OASIS CMIS specification on page 148

Content streams

The nonformatted data from an IBM Content Manager OnDemand document is retrieved in a byte array from the first content stream in the CMIS document.

In IBM Content Manager OnDemand, documents that are large objects have data in multiple segments. However, in IBM CMIS for Content Manager OnDemand, all of the segments are put into the first content stream. The segments cannot be mapped to individual content streams because segments have pages and there is no mechanism to retrieve individual pages in a segment.

Parent topic: IBM CMIS for Content Manager OnDemand implementation of the OASIS CMIS specification on page 148

Versioning

IBM Content Manager OnDemand does not support document versions, so the cmis:versionable attribute will always return false. The getAllVersions method will always return a single version. The checkout and checkin methods will issue notSupported exceptions or return empty lists if the method cannot issue this exception.

Parent topic: IBM CMIS for Content Manager OnDemand implementation of the OASIS CMIS specification on page 148

Folders

IBM Content Manager OnDemand does not have folders (OASIS CMIS folders that are used for filing documents). Therefore, IBM CMIS for Content Manager OnDemand implements the minimum folder requirements of the OASIS Content Management Interoperability Services specification.

You cannot use the following methods to retrieve folders:

- The getFolderParent method throws an invalidArgument exception for the root folder. The method returns a null value if any other folder ID is specified.
- The getObjectParent method always throws a constraint exception because IBM Content Manager OnDemand documents are not filed in folders.

Parent topic: IBM CMIS for Content Manager OnDemand implementation of the OASIS CMIS specification on page 148

Paging

The IBM CMIS for Content Manager OnDemand implementation for paging a list of query results is not efficient. Therefore, you should use queries that do not return a large number of results. If a query contains a large number of results, the performance of the IBM CMIS for Content Manager OnDemand server can be impacted.

Parent topic: IBM CMIS for Content Manager OnDemand implementation of the OASIS CMIS specification on page 148
Full text search

IBM Content Manager OnDemand does not expose full text searching through the ODWEK API. Therefore the cmis:fulltextIndexed property returns a value of false.

Parent topic: IBM CMIS for Content Manager OnDemand implementation of the OASIS CMIS specification on page 148

Query limitations

IBM Content Manager does not support all of the query operations that are supported by OASIS Content Management Interoperability Services.

IBM CMIS for Content Manager OnDemand throws an exception if a client application submits a request for one of the following query operations:

FROM clause
   Throws an exception if the query includes anything other than the cmis:document method.

IN_FOLDER operation query
   IBM Content Manager OnDemand does not have folders.

IN_TREE operation query
   IBM Content Manager OnDemand does not have hierarchical folders.

JOIN clause query.
   IBM Content Manager OnDemand only allows you to search against the cmis:document type.

CONTAINS operation query
   IBM Content Manager OnDemand does not support full text searching.

SCORE operation query.
   IBM Content Manager OnDemand does not support full text searching.

ANY operator query
   IBM Content Manager OnDemand does not support properties with multiple values.

Stand-alone negation operator query NOT
   Allowed only as part of other operators, such as NOT LIKE. The NOT operator cannot be used to negate an entire search clause, for example, NOT (prop > 5 AND prop < 100)

Combination of the AND and OR Boolean operator queries
   IBM Content Manager OnDemand supports only all AND operators or all OR operators.

Parentheses when used in conjunction with a combination of AND and OR operator queries
   IBM Content Manager OnDemand Web Enablement Kit does not support parentheses.
Notices

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

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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

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

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Portions of this product are:

- Copyright © 1991, 2012 Oracle and/or its affiliates. All rights reserved.

The Oracle Outside In Technology included herein is subject to a restricted use license and can only be used in conjunction with this application.

"Privacy policy considerations"

"Trademarks"

---

**Privacy policy considerations**

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

Depending upon the configurations deployed, this Software Offering may use persistent cookies that collect each user’s user name for purposes of enhanced user usability. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

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


---

**Trademarks**

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

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

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

Parent topic: Notices on page 155
Index

Special characters

| feature icons  | creating SVG files | 34 |
| icons          | creating SVG files | 34 |
| policies and security | 151 |

A

Access Control Lists (ACls)

| permission mappings for IBM CMIS for Content Manager | 140 |
| permission mappings for IBM CMIS for FileNet Content Manager | 113 |

actions

| creating plug-ins | 29 |
| creating menu actions | |
| creating plug-ins for administration tool widgets | 29 |
| desktops | 88 |
| graphical interface | 88 |
| labels | 88 |
| menus | 88 |

APIs

| reference | 101 |

APIs

| IBM Content Navigator | 1 |
| OASIS CMIS specification | 103 |

applications

| creating features | 32 |
| creating layouts | 35 |
| creating menu actions | 29 |
| creating menus | 30 |
| creating plug-in components | 28 |
| creating plug-ins | 25 |
| creating repository types | 36 |
| creating request or response filters | 38 |
| creating services | 40 |
| creating viewers | 41 |
| creating widgets | 43 |
| developing with IBM Content Navigator | 1 |
| development architecture | 15 |
| midtier services | 17 |
| packaging plug-in components | 44 |
| plug-in | 19 |
| sample custom viewer | 19 |
| sample external data service | 19 |
| sample web pages | 19 |

architecture

| development | 15 |
| midtier services | 17 |

B

bytes in HTTP header

| range | 113 |

© Copyright IBM Corp. 2012, 2014

159
layout widgets
  ecm.widget.layout package 93
layouts
  creating plug-ins for 35
  limitations 121
cmis:changeToken property 109
paging of feeds 110, 119
private working copy links 110, 118
queries 110, 119
listView widget library
  content list widgets package 89

mapping 135
  document type and folder type 135
  properties 137
menu widgets
  administration tool 88
menus
  creating plug-ins for 30
message widgets
  dialog boxes 90
  model updates 133
  modeling library
  structure 75
multiple items 121

OASIS CMIS APIs
  developing with 103, 116
  best practices 115
OASIS CMIS specification
  description 103
  error responses request
  object types request
  external data service 55
  GET method 55
  GET method example 57
  response content 62
ObjectStoreDBType
  See extensions
ObjectStoreID
  See extensions

p
  paging 142, 143
  particular object type resource example 69
  external data service 57
  POST method 58
plug-in components
  creating 28
  features 32
  layouts 35
  menus actions 29
  menus 30
  packaging 44
  repositories 36
  request or response filters 38
  services 40
  viewers 41
  widgets 43

plug-ins
  creating 25
POST method
  particular object type resource 58
process widgets
  ecm.widget.process package 95
property mapping 137
property type mapping 137, 138

RecordsManagementEnabled
  See extensions
repository types
  creating plug-ins for 36
  request filters
  creating plug-ins for 38
  request modes
    finalExistingObject 61
    finalNewObject 61
    initialExistingObject 61
    initialNewObject 61
    inProgressChanges 61
  requestMode parameter
    particular object type resource 61
  response filters
    creating plug-ins for 38
    restricted characters 121

sample applications
  customizing 19
  external data service 19
  plug-in 19
  viewer 19
  web pages 19
sample external data service
  customization choosing lists 47
search criteria
  implementing dependent choice lists 53
search widgets
  dialog boxes 90
  ecm.widget package 79
  ecm.widget.search package 96
  selection widgets
  dialog boxes 90
  service collection URLs 101
  service documents 101
  URI templates 101
services
  creating plug-ins for 40
  SVG files
    feature icons 34

teamspace builder widgets
  ecm.widget.teamspaceBuilder package 98

teamspace widgets
  ecm.widget package 79
TextSearchEngine
  See extensions

U
URLs
  creating for IBM Content Navigator 23
  custom search form 23
  custom search results 23
  features 23
  shortcut URLs to worklists and in-baskets 23
user interface widgets
  ecm.widget package 79
user widgets
  ecm.widget package 79

V
version policies 147
versioning 147
viewers
  creating plug-ins for 41
visual widget library
  administration model widgets
    package 89
  administration widgets package 88
  dialog box widgets package 90
  layout widgets package 93, 96
  process widgets package 95
  teamspace builder widgets package 98
  viewer widgets package 98
  widgets package 79

W
widgets
  administration model widget classes 89
  administration widget classes 88
  content list widget classes 89
  creating plug-ins for 43
  dialog box widget classes 90
documents 79
folders 79
layout widget classes 93
process widget classes 95
search widget classes 96
searches 79
teamspace builder widget classes 98
teamspaces 79
users 79
viewer widget classes 98
widget classes 79
workflows 79
work item widgets
  dialog boxes 90
workflow widgets
  dialog boxes 90
ecm.widget package 79
working with 120

Index 161
Product Numbers: 5697-H60  
5697-N93  
5724-B19  
5724-J33  
5724-R81  
5770-RD1