IBM Case Manager: Converting REST API calls to Model API calls

Type of Submission: Article
Title: IBM Case Manager
Subtitle: Converting REST API calls to Model API calls

Keywords: ICM, Case Manager, iWidgets, dojo

Prefix: Dr.
Given: Tim
Middle:
Family: Morgan
Suffix:

Job Title: ICM Enablement Architect
Email: tim@us.ibm.com
Bio: Dr. Morgan holds a Ph.D. in Computer Science from the University of California. He started with FileNet in 2004 as a Performance Analyst, and he developed the System Manager Dashboard performance monitoring framework. For several years, he has been the Software Architect for Enterprise Content Management HA/DR and Performance. Recently he was named to the IBM Case Manager Enablement architect role.
Company: IBM
Photo filename: Picture Small.jpg

Abstract: This article describes the framework differences between the Case REST API and the Model API used in Case Manager 5.2. It provides a mapping from Case REST calls to equivalent Model API calls for those converting existing widgets to Case Manager 5.2.

Introduction

Most custom widgets developed for use with IBM Case Manager V5.1 make use of one or more REST APIs. There are three APIs provided by IBM, the IBM CMIS for FileNet Content Manager API, the Process Engine REST Service, and the IBM Case Manager REST API for performing case-related operations that are not available through the CMIS and Process Engine REST APIs.

These APIs are still available in IBM Case Manager V5.2 for backward compatibility with the prior release. However, the framework provided by IBM Content Navigator provides higher level interfaces for content repositories and workflows that are independent of the underlying servers. IBM Case Manager V5.2 adds a new Case Manager Model API that works in a compatible fashion and serves the same purpose as the IBM Case Manager REST API does in prior releases of IBM Case Manager. This "Model Layer API" approach also offers a number of other advantages that are discussed in this article. For this reason, it is strongly recommended that custom widgets being converted from IBM Case Manager V5.1 to V5.2 also be modified to take advantage of the Model Layer API approach in place of continuing to use the REST APIs.

These APIs have different paradigms of operation, and different types of objects are returned as results from the requests. This document will contrast the two approaches and highlight the changes that a developer will need to make to convert REST calls to Model API calls, in the process of converting a widget to IBM Case Manager V5.2.

The REST paradigm

Of the three REST APIs being discussed, the CMIS API is platform-agnostic, while the Process Engine and Case Manager REST APIs are product-specific. Otherwise, all three function in the same way. At a low level, the client performs one of the four HTTP operations, GET, PUT, POST, or DELETE, against a specific URL. The URL encodes any parameters needed for the operation, either as elements of the URL path, or as parameters passed after a question mark (?) at the end of the path. Depending on the particular operation, PUT, POST, and DELETE actions may take additional parameter data in the request body, in JSON format.

As an example, to get a list of case types defined in a solution, a client can issue a GET request against a URL of the form:

http://*host:port*/*context*/CASEREST/v1/*resourceName*
[?TargetObjectStore=*objectStoreName*][&{*resourceParameters*}]

The server returns a list of objects describing the defined case types, in JSON format.

To keep the browser user interface responsive, JavaScript applications normally do not block waiting for the response from such a call. Instead, the call is made asynchronously, and a function is invoked with the response once it is received. An error function is invoked if the call fails or times out. JavaScript libraries all provide parameterized functions for accomplishing such calls. In Dojo, the developer creates a request object, then calls dojo.xhr(operation, request) to start processing the request asynchronously. The request object specifies the functions that should be invoked upon successful completion or if an error is encountered. The operation parameter is one of "GET", "PUT", "POST", or "DELETE".

Once a response is received, the JavaScript code might convert the JSON object into a native JavaScript object using functions such as dojo.fromJson(). The response object can then be processed easily in the subsequent JavaScript code.

More than one widget on a page can display data from a particular object. Suppose, for example, that an In-basket widget is displaying a list of work items. The user selects one of the items, which brings up that work item in a separate widget. Using that widget, the user changes the work item to complete a step for example. After that update is processed by the server, the In-basket widget must be updated in order to show the new state of the work item or to remove it from the list altogether. In IBM Case Manager V5.1.1 and earlier versions, this update is effected by publishing an event whose name is indicative of what has happened (workItemUpdated, for example).

The Model API paradigm

The Model API pattern originated with the IBM Content Navigator product. It allows developers to write client code, as well as the user interface code of IBM Content Navigator itself, that is independent of the underlying repositories, workflow systems, and so on, that IBM Content Navigator provides an interface to. REST APIs, in contrast, are more tied to particular products, although the CMIS REST API can be used against other CMIS-compliant repositories.

Another key difference is that the result of making a Model call is a Model object that represents the result. Unlike the result object from a REST call, the Model object has methods as well as data members. All IBM Content Navigator Model objects reside in the ecm.model namespace. For example, the ecm.model.Repository object has a method to add (create) a folder within the repository. It takes a number of arguments, including an object representing the parent folder for the new folder, the object store where the folder is to be created (for P8), permission information, and others. As with the REST APIs, Model API operations are performed asynchronously. The last argument to each API call is therefore a "callback" function that will be invoked when the operation completes. In the case of the create folder operation, that function receives an

ecm.model.ContentItem object that is the new folder object. That object, in turn, can be used to add documents to that folder.

As with the Case REST API, Case Manager defines an API for doing case-specific operations that extends the Content Navigator APIs. All of the Case Manager Model API objects reside in the icm.model namespace. These objects, detailed below, allow you to enumerate deployed solutions, create cases, enumerate the tasks associated with a case, and so on.

In some cases, the Case Manager Model API extends the IBM Content Navigator API by adding additional functionality to an existing object. For example, icm.model.WorkItem is used in some cases instead of ecm.model.WorkItem. The icm.model.WorkItem object contains case-specific methods and data, such as getCaseType(), which returns an icm.model.CaseType object. In other cases, the IBM Content Navigator objects are used natively. Roles in Case Manager Client, for instance, are represented as ecm.model.ProcessRoles objects. You can convert an ecm.model.WorkItem to an icm.model.WorkItem object using the icm.model.WorkItem.fromWorkItem() method.

Mapping REST calls to the Model API

In Case Manager Client, there is an icm.model._DesktopMixin object that mixes in the IBM Content Navigator Desktop object and adds case-specific functionality to it. So documentation for methods specific to IBM Case Manager is found in the JSDocs for icm.model._DesktopMixin, but users actually use ecm.model.desktop to call the methods. Users do not construct any of these objects directly, but receive them by calling various API calls. For example, the retrieveSolutions() call for the mixin returns a list of icm.model.Solution objects, that can then be used in various ways.

Getting information about solutions deployed

Operation	Model API
Get list of solutions	ecm.model.desktop.retrieveSolutions()
Get list of the document types that are defined in a solution	icm.model.Solution.retrieveDocumentTypes()
Get information about a particular solution	Various methods and fields defined in icm.model.Solution
Get list of cases	Use the IBM® Content Navigator API to search for Case objects. From a result item returned from a search (a ContentItem object) you can obtain a Case object using the Case.fromContentItem method.

Operation	Model API
Query in-basket	You can also use the IBM Content Navigator API to perform in-basket queries. From a result item returned by an in-basket query (an ecm.model.WorkItem object) you can obtain an icm.model.WorkItem object by using the WorkItem.fromWorkItem method.

Getting information about deployed case types

Operation	Model API
Get list of case types	icm.model.Solution.retrieveCaseTypes()
Get list of view definitions	There is no equivalent of this call in IBM Case Manager V5.2. There are two views remaining from IBM Case Manager V5.1.1 defined, searchView and summaryView, and they are both available as fields in the icm.model.CaseType object. Their values are similar to the equivalent Case REST values. In addition, a CaseType can have a default view, available as the CaseType.defaultViewDefinition field. CaseType supports two related API calls, getViewDefinitionResourceBundle() and getViewDefinitionUri(), that are primarily used by the Properties widget. They take a view definition ID as their arguments, not the objects that searchView and summaryView contain.
Get list of discretionary task types	icm.model.CaseType.retrieveDiscretionaryTaskTypes()
Get a particular case type	IBM Case Manager V5.2 follows Content Navigator in the use of the word “attribute” to refer to Case and WorkItem properties. Call icm.model.CaseType.retrieveAttributeDefinitions() to retrieve the generic attributes for a Case Type, or caseObj.retrieveAttributes() to retrieve the attributes unique to a specific case instance.
Get case page	icm.model.CaseType.retrievePage()

Getting and changing case information

Operation	Model API
Cases resource---create a case	icm.model.CaseEditable.createPendingInstance(). Then set appropriate values then call the new object's save() method. Note that the object store specification is inherited from the CaseType object argument, which in turn inherits it from the Solution object that the CaseType object came from, so it is not specified explicitly as in the REST API. Two other related functions are available under icm.model.CaseEditable: • createPendingSplitInstance() creates a CaseEditable object that represents a new Case object that is split from another Case. • fromCaseObject() constructs a CaseEditable object and associates with an existing Case object.
Get particular case instance	The various methods and fields of the icm.model.Case object are used to access case instance data. The icm.model.Case object can be obtained by calling icm.model.Solution.retrieveCase() or icm.model.Case.fromContentItem(). The latter would be appropriate if the query were performed through the IBM Content Navigator API.
Update particular case instance	icm.model.Case.createEditable(), update values on the returned CaseEditable object, then call save()
Status of particular case	You can get the case status as one of the properties (“attributes”) of the case. From a Case object you can access caseObj.attributes["CmAcmCaseState"]. Unlike the REST API call, the value for the CmAcmCaseState property is an integer value. A choice list defined in the Content Platform Engine maps these values to the symbolic names used in the REST API. For example, the value 2 is “Working”.
Related cases for a particular case	icm.model.Case.retrieveRelatedCases(). To change the relationship between cases, call icm.model.Case.relateCase() or icm.model.Case.unrelateCase().

Operation	Model API
List of task instances	icm.model.Case.retrieveTasks()
Create new task	icm.model.TaskEditable.createNew(), set values then call save()
Particular task instance (change the state of a task: enable, disable, start, stop, or restart)	icm.model.Task.enable(), disable(), start(), stop(), or restart().
Get Case comments for a Case, Task, Document, or WorkItem	icm.model.Case.retrieveCaseComments(), icm.model.Case.retrieveTaskComments(), icm.model.Case.retrieveDocumentComments(), or icm.model.Case.retrieveWorkItemComments(),
Put (create) Case comments on a Case, Task, Document, or WorkItem	icm.model.Case.addCaseComment(), addTaskComment(), addDocumentComment(), or addWorkItemComment()
Case history	icm.model.Case.searchHistory()

Search and the SearchTemplate class

In IBM Case Manager V5.2, searches are performed leveraging the repository-independent search capabilities provided by Content Navigator. See Migrating a Custom Search Widget from ICM 5.1.1 to ICM 5.2 for details of how to perform a search in IBM Case Manager V5.2.

Conclusion

When converting widgets from a V5.1.1 environment to a V5.2 environment, replace the REST calls with calls to the IBM Case Manager V5.2 Model API. This article describes the mapping between the Case REST API calls and the corresponding Model API. For the details of each Model API call, please refer to "IBM Case Manager JavaScript API References" in the IBM Case Manager Information Center using the link provided in the References section below.

References

- The Case REST Protocol: http://pic.dhe.ibm.com/infocenter/casemgmt/v5r1m1/topic/com.ibm.casemgmt.development.doc/acmdv014.htm

- IBM Case Manager JavaScript icm.model package: http://pic.dhe.ibm.com/infocenter/casemgmt/v5r0m0/topic/com.ibm.casemgmt.development.doc/acmjs004.htm
- Migrating a Custom Widget from ICM 5.1.1 to ICM 5.2: https://www.ibm.com/developerworks/mydeveloperworks/blogs/e8206aad-10e2-4c49-b00c-fee572815374/resource/ACM_LP/ICM52MigrateCustomWidgets.pdf