IBM Case Manager
Version 5.3.3

Development Guide

SC19-3682-09

IBM Case Manager
Version 5.3.3

Development Guide

SC19-3682-09

Note

Before using this information and the product it supports, read the information in "Notices" on page 173.

This edition applies to Version 5 Release 3 Modification 3 of IBM Case Manager (product number 5725-A15) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Developing case management applications

IBM® Case Manager and the IBM FileNet® P8 software provide tools for building custom web applications to manage cases. You can use various extension points and application programming interfaces (APIs) to extend Case Manager Client by adding custom pages, widgets, actions, events, or services. Alternatively, you can use the APIs to build custom applications that incorporate IBM Case Manager features without using Case Manager Client.

You can use Case Manager Builder to create a solution by using one of the industry-solution templates provided by IBM or by using a blank template. You can then modify that solution to meet your requirements.

Restriction: You cannot use a custom application to create a solution. You must use Case Manager Builder to create or modify solutions.

You might create an application that implements a custom widget that interacts directly with the Case Manager Client widgets. For example, if a Work Item Toolbar widget does not provide the functionality you need, you can implement a custom widget to replace that widget. You can then wire the custom widget with the other Case Manager Client widgets on the Work Details page and the Add Task page.

Alternatively, you might create an application that enables case workers to process cases without using Case Manager Client. For example, you might create such an application to use your existing user interface for processing cases with IBM Case Manager.

You can use the following APIs to extend your case management client application:

IBM Case Manager JavaScript API
Use this API to customize your Case Manager Client application.

IBM Case Manager Java™ API
Use this API to create servlets for custom web applications and to develop custom component queue applications.

Content Platform Engine add-on extensions for Case Manager Builder
Use these add-on extensions to access the custom metadata and data that is stored in object stores.

IBM CMIS for FileNet Content Manager
Use the IBM Content Management Interoperability Services (CMIS) for FileNet Content Manager to enable applications and clients that use the OASIS CMIS standard to access content that is stored on Content Platform Engine. In a case management application, you use CMIS to manage the case folders and documents and to retrieve case information.

Developing case management applications with the JavaScript API

IBM Case Manager provides a JavaScript application programming interface (API) that you can use to customize your case management client application. For example, you can use this API to create cases, gather information about solutions, and start manual tasks.

The IBM Case Manager JavaScript API provides classes that represent case management objects such as solutions, case types, cases, and tasks. It also provides classes that represent components of the client user interface such as page widgets, dialog boxes, toolbars, and pop-up menus.

The IBM Case Manager JavaScript API uses the Dojo toolkit, which is an open source JavaScript library for web development.

In addition to the IBM Case Manager JavaScript API, you use the IBM Content Navigator JavaScript API to customize your client application. The IBM Content Navigator JavaScript API includes more modeling classes and widget classes that you can use in your application.

"IBM Case Manager JavaScript packages"

Related information:

IBM Content Navigator JavaScript API Reference

IBM Case Manager JavaScript packages

The classes in the IBM Case Manager JavaScript API are divided into packages based on functionality.

"IBM Case Manager JavaScript `icm.model` package"
"IBM Case Manager JavaScript `icm.action` package" on page 7
"IBM Case Manager JavaScript `icm.base` package" on page 12
"IBM Case Manager JavaScript `icm.dialog` package" on page 12
"IBM Case Manager JavaScript `icm.pgwidget` package" on page 13
"IBM Case Manager JavaScript `icm.util` package" on page 14
"IBM Case Manager JavaScript `icm.widget.menu` package" on page 14

IBM Case Manager JavaScript `icm.model` package

The classes in the `icm.model` package contain the classes that represent the objects in the case management domain. These case management objects, which include solutions, cases, work items, and tasks, map to Content Platform Engine objects on the server.

Case Manager Client is a plug-in to IBM Content Navigator. The IBM Content Navigator model provides the capabilities for searching and retrieving Content Platform Engine objects on the server. However, this model lacks the semantics for the case management context. Therefore, the IBM Case Manager model provides the mechanism for adding the context that is needed for users to search and retrieve case management objects.

The IBM Case Manager model classes are derived from the base classes in the IBM Content Navigator JavaScript model. In addition, some IBM Case Manager model classes use the functionality of IBM Content Navigator classes. For example, an `WorkItem` object is obtained by using an `ecm.model.WorkItem` object.

Each IBM Case Manager model class defines methods for creating, reading, updating, and deleting the case management object that it represents.

Important: Although you can use the model API to create the artifacts in a solution, you cannot use the API to create a solution. Instead, you must use Case Manager Builder to create the solution.

A model class also defines methods that enable the object to reference related case management objects. For example, the `Case.retrieveTasks()` method is used to fetch `Task` objects that are related to a case. In many situations, a widget uses the model objects received in an event payload to navigate the model API and retrieve needed information. This ability simplifies the data that a widget must pass in events because a widget can pass a model object instead of passing all the information for that object. For example, an event to open a work item can pass a `WorkItem` object or a `WorkItemEditable` object in the payload. If this work item event is received by a milestones widget, that widget can call the appropriate model API method to retrieve the milestones. The originating widget does not need to collect and pass the milestone information.

For certain objects, the model defines two related classes. The persistent class represents the object as it is saved in the object store. The other class represents an editable version of the object. This editable class, sometimes called a *scratchpad*, represents the object as it is being edited.

The widgets on a page share an editable object to enable the widgets to coordinate changes to the object. Events notify the widgets to any changes in the editable object. For example, the `caseEditable.onRefresh()` event is called when

- The `CaseEditable` object is saved.
- A `CaseEditable` object that was obtained from the same `Case` object is saved.

For editable properties, the `onChange()` methods, such as `onChoiceListChanged` and `onValueChanged`, notify the widgets when updates are made to the editable object. The changes that are made to the editable object are saved to the corresponding persistent object only when the user saves the page.

The editable object is shared only by widgets that are on the same page. If multiple pages related to the same object are open, each page has its own editable object. However, the model is defined so that when a user saves changes to the editable object on one page, the editable objects on the other pages are refreshed. The widgets on these pages are notified of changes by listening for the event that is triggered by the onRefresh() method on the editable object.

Table 1. Classes in the `icm.model` package

Persistent class	Editable class	Description
`Case`	`CaseEditable`	Represents a case. To obtain a `CaseEditable` object to create a case, call the `createNewCaseEditable` method on the `Solution` object. To obtain a `CaseEditable` object to edit an existing case, call the `createEditable` method on the `Case` object. The `propertiesCollection` attribute of the `CaseEditable` class provides a collection of `PropertyEditable` objects. Each `PropertyEditable` object represents a property value for a case.
`CaseComment`		Represents a comment that is entered for a case.
`CaseRelationship`		Represents the relationship between two cases.
`CaseType`		Represents a case type.
`DocumentType`		Represents a document class.
`HistoryEvent`		Represents the record of an event in a case history.
`InbasketDynamicFilter`		Represents a dynamic filter type.
`InbasketFilter`		Represents a inbasket filter type.
None	`PropertyEditable`	Represents a property of a case or a parameter of a launch step or work item.
`ResultSet`		Represents a set of search results or other items that are returned by a query to the content server.
`Solution`		Represents a solution.

Table 1. Classes in the `icm.model` package (continued)

Persistent class	Editable class	Description

`Task`	`TaskEditable`
	
	`LaunchStep`
		Represents a task.
		
		`TaskEditable` objects are primarily used to represent new discretionary tasks. To obtain a `TaskEditable` object for a discretionary task, call the `createNewTaskEditable` method on the `Case` object.
		
		To obtain a `TaskEditable` object to edit an existing task, call the `createEditable` method on the `Task` object.
		
		For discretionary tasks, the model includes the `LaunchStep` class that represents the launch step of a workflow. The `propertiesCollection` attribute of this class provides a collection of `PropertyEditable` objects. Each `PropertyEditable` object represents a parameter for the launch step.

`TaskType`		Represents a task type in a deployed case management solution.
`Timeline`		Represents a timeline object.
`TimelineEvent`		Represents a timeline event of a given case.
`TimelineOverview`		Represents a timeline overview of a given case.
`TimelineTask`		Represents a timeline task of a given case.
`TimelineWorkitem`		Represents a timeline work item of a given task.
`WorkItem`	`WorkItemEditable`	Represents a work item. To obtain a `WorkItemEditable` object, call the `createEditable` method on the `WorkItem` object. The `propertiesCollection` attribute of this class provides a collection of `PropertyEditable` objects. Each `PropertyEditable` object represents a parameter for the work item.

The following classes are included in the `icm.model.properties.controller` package:

`icm.model.properties.controller.ControllerManager`
Represents a manager that is used to retrieve the property controllers for editable objects.

icm.model.properties.controller.PropertyCollectionController
: Represents a collection of property controllers that are bound to the properties of an editable object.

icm.model.properties.controller.types._PropertyController
: Provides the base controller class for a property.

icm.model.properties.controller.types.AttachmentPropertyController
: Represents the controller for a property of type `attachment`.

icm.model.properties.controller.types.BooleanPropertyController
: Represents the controller for a property of type `boolean`.

icm.model.properties.controller.types.DatetimePropertyController
: Represents the controller for a property of type `datetime`.

icm.model.properties.controller.types.FloatPropertyController
: Represents the controller for property of type `float`.

icm.model.properties.controller.types.GroupPropertyController
: Represents the controller for a property of type `group`.

icm.model.properties.controller.types.IntegerPropertyController
: Represents the controller for property of type `integer`.

icm.model.properties.controller.types.StringPropertyController
: Represents the controller for a property of type `string`.

IBM Case Manager JavaScript `icm.action` package

The classes in the `icm.action` package represent that actions that users can perform on case management objects. You can add these actions to the toolbars or pop-up menus for a widget.

A IBM Case Manager action works for a specific type of object such as a case or a work item. The object is indicated by the subpackage within the `icm.action` package. One exception is the `icm.action.util` package, which contains actions that are not performed for specific case management objects.

An action also requires a specific context within which the action works. The context identifies the objects that the action requires. The context also determines on which toolbars and menus the action is available. For example, the action that is represented by the `icm.action.case,AddCustomTask` class requires either an `icm.model.Case` object or an `icm.model.WorkItem` object. This action is available on the toolbar for editing a case or opening a work item. The action is not available on the toolbar for adding a case or work item.

In the following tables, the context for each action is shown in single or double brackets:

Table 2. Context syntax

Syntax	Description
`['Case', 'WorkItem']`	The action requires either a case object or a work item object.
`[['NewCase', 'Coordination']]`	The action requires a new case object and a coordination object.
`[['CasePage', 'Coordination'], ['NewCase', 'Coordination']]`	The action requires either a case page object and a coordination object or a new case object and a coordination object.

`icm.action.attachment` package

The `icm.action.attachment` package defines a single class, Remove, for the attachment context. A `Remove` object is used to remove a document from an attachment.

`icm.action.case` package

The `icm.action.case` package defines actions that are performed for cases.

Table 3. Classes in the `icm.action.case` package

Class	Context	Description
AddCaseAndClosePage	`[['NewCase', 'Coordination']]`	Saves the case that is being added, and then closes the current Add Case page or Split Case page.
AddCustomTask	`['Case', 'WorkItem']`	Opens the Custom Task Editor window so that the user can add a custom task to a case.
AddCustomTaskFromExisting	`['Case', 'WorkItem']`	Opens a copy of the selected task in the Custom Task Editor window so that the user can add a new custom task to a case.
CloseCasePage	`[['CasePage', 'Coordination'], ['NewCase', 'Coordination']]`	Closes the current Add Case page, Case Details page, or `Split Case` page without saving any changes.
OpenAddPredefinedTaskPage	`['Case', 'WorkItem']`	Adds a discretionary task to the case.
OpenCasePage	`['CaseReference']`	Opens the selected case in the Case Details page.
OpenSplitCasePage	`[['Solution', 'Case']]`	Opens the Split Case page so that the user can reuse properties from an existing case to create a new case.
SaveCaseOnPage	`[['CasePage', 'Coordination'], ['NewCase', 'Coordination']]`	Saves the case that is being edited or added without closing the page.
SendLink	`['Case']`	Sends an email that contains the URL to open the selected case in the Case Details page.
ShowLink	`['Case']`	Displays the URL to open the selected case in the Case Details page.

`icm.action.comment` package

The `icm.action.comment` package defines actions that are used to add comments to cases, documents, tasks, and work items.

Table 4. Classes in the `icm.action.comment` package

Class	Contenxt	Description
AddCaseComment	`['Case']`	Opens the Comments window so that the user can add a comment or view comments for a case.

Table 4. Classes in the `icm.action.comment` package (continued)

Class	Contenxt	Description
AddDocumentComment	`[['Case', 'Document']]`	Opens the Comments window so that the user can add a comment or view comments for a document.
AddTaskComment	`['WorkItem', 'Task']`	Opens the Comments window so that the user can add a comment or view comments for a task.
AddWorkItemComment	`['WorkItem']`	Opens the Comments window so that the user can add a comment or view comments for a work item.

`icm.action.contentitem` package

The `icm.action.contentitem` package defines actions that are performed for documents and folders.

Table 5. Classes in the `icm.action.contentitem` package

Class	Context	Description
Cut	`['Document']`	Removes the selected document from the case.
Open	`['Folder', 'Document']`	Opens the selected document or folder.
Paste	`['CurrentFolder']`	Pastes a document into the case.

`icm.action.document` package

The `icm.action.document` package defines actions that are performed for documents.

Table 6. Classes in the `icm.action.document` package

Class	Context	Description
AddDocumentFromLocal	`['CurrentFolder']`	Adds a document to a case or a case folder. When **Allow documents and attachments from repositories other than the case management object stores** is selected, documents can be saved directly to a case without selecting a repository or folder.
Open	`['Document']`	Opens the selected document.
Refresh	`['Document']`	Refreshes the document.

`icm.action.folder` package

The `icm.action.folder` package defines actions that are performed for folders.

Table 7. Classes in the `icm.action.folder` package

Class	Context	Description
AddFolder	`['CurrentFolder']`	Adds a folder to the case.

Table 7. Classes in the `icm.action.folder` package (continued)

Class	Context	Description
Open	`['Folder']`	Opens the selected folder and displays its content.

`icm.action.solution` package

The `icm.action.solution` package defines actions that are performed for solutions.

Table 8. Classes in the `icm.action.solution` package

Class	Context	Description
EditProcessPreferences	`['Solution']`	Opens the Preference window so that the user can edit notification preferences for processes.
ManageRoles	`['Solution']`	Opens the Manage Roles window so that the user can assign users to roles in a solution.
OpenAddCasePage	`['Solution']`	Opens the Add Case page so that the user can create a case of the selected case type.

`icm.action.task` package

The `icm.action.task` package defines actions that are performed for tasks.

Table 9. Classes in the `icm.action.task` package

Class	Context	Description
AddTaskAndClosePage	`[['NewTask', 'Coordination']]`	Starts the new task and closes the current Add Task page.
CancelAddTaskPage	`[['NewTask', 'Coordination']]`	Cancels the addition of a new task and closes the current Add Task page.

`icm.action.utility` package

The `icm.action.utility` package defines actions that are not related to specific case management objects.

Table 10. Classes in the `icm.action.utility` package

Class	Context	Description
EventAction	None	Creates a button or menu item that publishes or broadcasts a custom event.
OpenWebPage	None	Opens the specified website in a separate window.
ScriptAction	None	Creates a button or menu item that runs a custom script.

`icm.action.workitem` package

The `icm.action.workitem` package defines actions that are performed for work items.

Table 11. Classes in the `icm.action.workitem` package

Class	Context	Description
CloseWorkItemPage	`[['WorkItemPage', 'Coordination']]`	Closes the current Work Details page without saving any changes.
DispatchWorkItemAndClosePage	`[['WorkItemPage', 'Coordination']]`	Dispatches the current work item. If the next work item is not opened automatically, this action also closes the current Work Details page.
MoveToInbox	`[['WorkItem', 'Solution']]`	Moves the selected work item to the user's personal in-basket.
OpenNextWorkItemInPage	`[['WorkItemPage', 'Coordination']]`	Opens the next available work item in the same page when the user dispatches the current work item.
OpenWorkItemPage	`['WorkItemReference']`	Opens the selected work item in the Work Details page.
Reassign	`[['WorkItem', 'Solution', 'Role']]`	Reassigns the selected work item to another user. If the work item is open, this action first closes the work item.
ReturnToSender	`[['WorkItem', 'Solution']]`	Returns a work item to the in-basket it was most recently in. If the work item is open, this action first closes the work item.
SaveWorkItemOnPage	`[['WorkItemPage', 'Coordination']]`	Saves the work item that the user is editing without closing the Work Details page.

"Action contexts"

Action contexts

The context for an action corresponds to one or more IBM Case Manager or IBM Content Navigator model classes.

The following table identifies the class or classes that each context represents:

Table 12. Action context model classes

Context	Class
Attachment	ecm.model.ContentItem
Case	icm.model.Case icm.model.CaseEditable
CasePage	icm.model.Case icm.model.CaseEditable
CaseReference	icm.model.Case icm.model.CaseEditable
Coordination	icm.util.Coordination
CurrentFolder	ecm.model.ContentItem

Table 12. Action context model classes (continued)

Context	Class
Document	ecm.model.ContentItem
Folder	ecm.model.ContentItem
NewCase	icm.model.CaseEditable
	icm.model.TaskEditable
Role	ecm.model.ProcessRole
Solution	icm.model.Solution
Task	icm.model.Task icm.model.TaskEditable
WorkItem	icm.model.WorkItem icm.model.WorkItemEditable
WorkItemPage	icm.model.WorkItem icm.model.WorkItemEditable
WorkItemReference	icm.model.WorkItem icm.model.WorkItemEditable

IBM Case Manager JavaScript `icm.base` package

The classes in the `icm.base` package that support the definition of custom events, actions, page widgets, and constants.

Table 13. Classes in the `icm.base` package

Class	Description
_EventStub	Provides methods that can be used to publish events for a custom page widget.
BaseActionContext	Provides an interface for exchanging context information between a page widget and an action.
BasePageWidget	Provides methods that can be used to return the role context, solution context, and attributes for a page widget. In addition, this class provides methods that can be used to publish events for a custom page widget.
Constants	Provides a collection of constant variables that are used in IBM Case Manager.
WidgetAttributes	Represents the attribute values that are set for a page widget.

IBM Case Manager JavaScript `icm.dialog` package

The `icm.dialog` package contains classes that represent the dialog box boxes that are used in Case Manager Client.

Table 14. Classes in the `icm.dialog` package

Class	Description
icm.dialog.addcommentdialog. AddCommentDialog icm.dialog.addcommentdialog. djit.CommentContentPane	Represents the Add Comment dialog box that case workers use to add and view comments for cases, documents, work items, or tasks. The `CommentContentPane` class represents the user interface panel that is used by the Add Comment dialog box.
icm.dialog.addtaskdialog.AddTaskDialog	Represents the Add Task dialog box that case workers use to add discretionary tasks to a case.
icm.dialog.dynamictaskeditor. DynamicTaskEditorDialog	Represents the Custom Task Editor dialog box that case workers use to add custom tasks to a case.
icm.dialog.reassigndialog.ReassignDialog	Represents the Reassign Items dialog box that case workers use to reassign work items to other case workers.
icm.dialog.showlinkdialog.ShowLinkDialog	Represents the Show Link dialog box that case worker use to view, copy, or email the URLs to cases.

IBM Case Manager JavaScript `icm.pgwidget` package

The `icm.pgwidget` package contains classes that represent the page widgets that are provided byIBM Case Manager.

These classes define the methods that are used to handle the incoming events for the IBM Case Manager widgets. In addition, some of the classes define extension points that you can use to customize the behavior of the page widgets.

Table 15. Classes in the `icm.pgwidget` package by widget

Widget	Class
Attachments	icm.pgwidget.attachment.Attachment
Case Information	icm.pgwidget.caseinfo.CaseInfo icm.pgwidget.caseinfo.dijit.CaseInfoComponentContentPane
Case List	icm.pgwidget.caselist.CaseList icm.pgwidget.caselist.CaseListViewDetails icm.pgwidget.caselist.CaseListViewMagazine
Case Toolbar	icm.pgwidget.casetoolbar.CaseToolbar
Content List	icm.pgwidget.contentlist.ContentList
Form	icm.pgwidget.caseform.CaseForm
In-baskets	icm.pgwidget.inbasket.Inbasket
Instruction	icm.pgwidget.instruction.Instruction
Original Case Properties	icm.pgwidget.originalcase.OriginalCase
Process History	icm.pgwidget.processhistory.Processhistory

Table 15. Classes in the `icm.pgwidget` package by widget (continued)

Widget	Class
Properties	icm.pgwidget.properties.Properties
Script Adapter	icm.pgwidget.scriptadapter.ScriptAdapter
Search	icm.pgwidget.casesearch.CaseSearch
Select Case Documents	icm.pgwidget.caseselectdocument.CaseSelectDocument
Split Case Properties	icm.pgwidget.splitcase.SplitCase
Timeline Visualizer	icm.pgwidget.casevisualizer.CaseVisualizer
Toolbar	icm.pgwidget.toolbar.Toolbar
Viewer	icm.pgwidget.viewer.Viewer
Website Viewer	icm.pgwidget.websitedisplayer.WebSiteDisplayer
Work Item Toolbar	icm.pgwidget.workitemtoolbar.WorkitemToolbar

IBM Case Manager JavaScript `icm.util` package

The classes in the `icm.util` package provide support for multiple widgets.

Table 16. Classes in the `icm.util` package

Class	Description
Coordination	Represents an object that is used to coordinate the communication among widgets that are on a page.
SearchPayload	Represents the payload of the event that contains criteria for a case search. This event is published by the Search widget and handled by the Case List widget.
Util	Provides the `getResourceBundle` method that can be used for different types of objects.

IBM Case Manager JavaScript `icm.widget.menu` package

The classes in the `icm.widget.menu` package represent the pop-up menus and toolbars that are used with page widgets.

The toolbars and menus in IBM Case Manager are Dojo widgets that can be used with any page widget.

Table 17. Classes in the `icm.widget.menu` package

Class	Description
ContextualMenu	Represents a pop-up menu for a widget.
Menu	Provides the base class for pop-up menus and toolbars.
MenuManager	Provides methods for managing menus in Case Manager Client.
Toolbar	Represents a toolbar for a widget.

Developing case management applications with the Java API

IBM Case Manager provides a Java application programming interface (API) so that you can create custom applications. For example, you can create applications that create cases, gather information about solutions, and start manual tasks.

With the Java API methods, you can develop many case management operations in your own applications:

Deployed solutions
You can gather the following information:
- The workflow system connection point that a specific solution is configured to use
- The case types and document classes that are included in a deployed solution
- The IBM Content Manager host information for the object store where the solutions are deployed, if the object store is configured for IBM Content Manager integration

Deployed case types
You can gather the following information:
- Which discretionary task types are available to be created
- Which page views are configured for a case type

Cases You can create the following operations for cases:
- Create cases
- Update the properties of cases
- Split cases
- Create relationships between cases
- Retrieve the cases that are related to another split case or a related case
- Add and retrieve comments on cases
- Retrieve history about cases

Tasks You can create the following operations for tasks:
- Retrieve a list of the tasks of a case
- Start manual tasks
- Enable and disable tasks
- Stop and restart the workflow associated with a task
- Create discretionary tasks

"Configuring your environment to use the Java API" on page 16
"Configuring your environment to use the Content Engine Java API" on page 16
"Java API Components" on page 16
"Example: IBM Case Manager Java API Context" on page 18

Configuring your environment to use the Java API

To develop case management applications with the IBM Case Manager Java API, you must configure your system to use both the IBM Case Manager Java API and the Content Engine Java API.

About this task

To configure your development environment, you must add specific JAR files to your class path.

Procedure

To configure your development environment:

1. Find the installation directory that contains the JAR files that you use to run the API. The installation directory is `installation_directory/IBM/CaseManagement/CaseAPI/lib` where *`installation_directory`* is the directory where IBM Case Manager was installed.
2. Include each JAR file in this directory in the class path.
3. If your application integrates with IBM Content Manager, find the JAR files in the `CaseAPI/lib_cm8` directory on your development system and include each one in the class path.

Configuring your environment to use the Content Engine Java API

To develop case management applications with the IBM Case Manager Java API, you must configure your system to use the IBM FileNet Content Engine Java API.

About this task

The IBM FileNet P8 Information Center describes how to configure an environment for the IBM FileNet Content Engine Java API. For example, a typical approach to configuring the IBM FileNet Content Engine Java API to use the WSI transport is:

Procedure

1. Include `xlxpScanner.jar`, `xlxpScannerUtils.jar`, and `stax-api.jar` in the `CLASSPATH`.
2. Define a system property to reference the appropriate JAAS configuration file as follows: `-Djava.security.auth.login.config=C:\CE_API\config\jaas.conf.WSI`.

Java API Components

The IBM Case Manager Java API is organized into a set of major components that you use to build an application.

Case class

The `Case` class represents a case in the case management system.

To obtain an instance of a `Case` object, use one of the factory methods:

- `createPendingInstance`
- `fetchInstance`

A new object is first created in a pending state with a method such as `createPendingInstance`. The actual object is not created in the repository until the `save` method is called.

A factory method such as `fetchInstance` obtains an instance that represents an existing `Case` object. State information about the object, such as its list of properties, is fetched from the repository and maintained in the returned instance. A `Case` instance can also be obtained without a fetch operation by calling the `getFetchlessInstance` method. With this method, no call is made to the server to verify that the object exists in the repository, allows certain operations to be run in a more efficient manner, since the original fetch of the object is bypassed.

Once a `Case` instance is obtained, other methods can be called to run various operations on the object, such as modifying its properties, fetching the history of the case, or adding comments.

CaseMgmtObjectStore class

The `CaseMgmtObjectStore` class represents an object store that contains a deployed solution.

To obtain an instance of the `CaseMgmtObjectStore` class, use one of the factory methods:

- `fetchInstance`
- `getFetchlessInstance`

If an instance is obtained by calling `fetchInstance`, the method verifies that the identifier used to specify an object store does reference a valid object store. An exception is thrown if the object store is invalid. If `getFetchlessInstance` is called, no such verification occurs. However, an exception might be thrown later if a method is called that requires a reference to a valid object store.

All of the information contained in an instance of `CaseMgmtObjectStore` is managed by a cache that is internal to the IBM Case Manager Java API. The same information can be accessed whether the instance was obtained with or without a fetch operation, but the `fetchInstance` method runs only an initial check to ensure that the referenced object store is valid.

CaseType class

The `CaseType` class represents a case type that is part of a deployed solution.

To obtain an instance of `CaseType`, use one of the factory methods:

- `fetchInstance`
- `getFetchlessInstance`

If an instance is obtained by calling `fetchInstance`, the method verifies that the referenced object store and case type symbolic name represent a valid case type. The method throws an exception if the case type is invalid. If getFetchlessInstance is called, no such verification occurs. An exception might be thrown later if a method is called that requires that the identifiers used to reference the case type are valid.

Much of the information contained in a CaseType instance is managed by a cache that is internal to the IBM Case Manager Java API. All of the information can be retrieved, whether the instance was obtained with or without a fetch operation. However, the fetchInstance method runs only an initial check to ensure that the referenced case type is valid.

DeployedSolution class

The DeployedSolution class represents a deployed solution in the case management system.

To obtain an instance of DeployedSolution, use one of the factory methods:
- fetchInstance
- getFetchlessInstance

A list of deployed solutions can be obtained by calling the fetchSolutions method.

If a DeployedSolution instance is obtained by calling fetchInstance, the method verifies that the referenced object store is valid and that the solution name refers to a valid solution. If the object store or solution is invalid, then the method throws an exception. If getFetchlessInstance is called, no such verification occurs. However, an exception might be thrown later if another method is called that requires that the identifiers used to reference the object store is valid.

Much of the information contained in a DeployedSolution instance is managed by a cache that is internal to the IBM Case Manager Java API. All of the information can be retrieved whether the object was obtained with or without a fetch, but the fetchInstance method runs only an initial check to ensure that the referenced solution is valid.

Example: IBM Case Manager Java API Context

With the correct context, a UserContext can be established in the calling thread before calling IBM Case Manager Java API methods or directly calling IBM FileNet Content Engine Java API methods.

The general structure necessary to call the IBM FileNet Content Engine Java API includes establishing both a UserContext and a CaseMgmtContext. The UserContext is established by using the IBM FileNet Content Engine Java API, and the CaseMgmtContext is established by using the IBM Case Manager Java API. For example, in a stand-alone environment, the overall structure might look like:

```
P8ConnectionCache connCache = new SimpleP8ConnectionCache();
Connection conn = connCache.getP8Connection(CE_URI);
Subject subject =
    UserContext.createSubject(conn, USER_NAME,
                              PASSWORD, "FileNetP8WSI");
UserContext uc = UserContext.get();
uc.pushSubject(subject);
Locale origLocale = uc.getLocale();
uc.setLocale(1);
CaseMgmtContext origCmctx =
    CaseMgmtContext.set(
        new CaseMgmtContext(
            new SimpleVWSessionCache(), connCache()
        )
    );
try {
    // Code that calls the Case Java API or
    // directly calls the CE Java API
    ...
}
finally {
    CaseMgmtContext.set(origCmctx);
    uc.setLocale(origLocale);
    uc.popSubject();
}
```

If the application is running as an IBM WebSphere® Application Server (WAS) application, so that the user is already authenticated by the application server, the code might look like:

```
HttpServletRequest request;
P8ConnectionCache connCache =
    new HttpP8ConnectionCache(request);
VWSessionCache vwSessCache =
    new HttpVWSessionCache(request);
UserContext origUc = UserContext.get();
UserContext uc = new UserContext();
uc.setLocale(request.getLocale());
UserContext.set(uc);
CaseMgmtContext origCmctx =
    CaseMgmtContext.set(
        new CaseMgmtContext(vwSessCache, connCache)
    );
try {
    // Code that calls the Case Java API or
    // directly calls the CE Java API
    ...
}
finally {
    CaseMgmtContext.set(origCmctx);
    UserContext.set(origUc);
}

public class HttpP8ConnectionCache
    implements P8ConnectionCache {
    // A custom implementation of P8ConnectionCache
    // that caches Connection objects
    // in the HttpSession of the HttpServletRequest
}

public class HttpVWSessionCache implements VWSessionCache {
    // A custom implementation of VWSessionCache
    // that caches VWSession objects in the HttpSession
    // of the HttpServletRequest
}
```

Developing case management applications with the REST protocols

You can use the REST protocols to incorporate IBM Case Manager features in your custom application. You use the IBM Case Manager REST protocol to access case-specific objects. You use the Process Engine REST Service to access workflow-related aspects of tasks.

The REST protocols are provided for compatibility with previous releases of IBM Case Manager. If you are developing a new custom case management application, use the JavaScript APIs that are provided with IBM Case Manager and IBM Content Navigator.

“Creating and managing case objects by using the IBM Case Manager REST protocol”

“Managing workflows, roles, and in-baskets by using the Process Engine REST Service” on page 93

Creating and managing case objects by using the IBM Case Manager REST protocol

You use the IBM Case Manager REST protocol to access and manipulate case-specific objects, including solutions, case types, tasks, case comments, and case histories.

“Case management REST resource URIs”

“Symbolic names” on page 22

“Error responses” on page 23

“Common JSON payload for cases and case types” on page 23

“Getting information about deployed solutions” on page 31

“Getting information about deployed case types” on page 37

“Getting and changing case information” on page 52

Case management REST resource URIs

Each resource in the IBM Case Manager REST protocol is identified by a unique URI. This URI includes the resource name and any parameters that are required for the specific method that you are calling.

Syntax

The URI syntax for case management REST resources is as follows:

```
http://host:port/context/CASEREST/v1/resourceName[?resourceParameters]
```

The variables used in the URI are:

host
The name of the server that is hosting the IBM Case Manager REST protocol as configured for your web application server.

port
The host port that is used for IBM Case Manager REST protocol communications as configured for your web application server.

context
The web application context root for the IBM Case Manager REST protocol as configured for your web application server. By default, this value is set to `CaseManager`.

resourceName
The name of the IBM Case Manager REST resource to use.

resourceParameters
Any parameters specified for the specified IBM Case Manager REST resource. Use an ampersand (&) to separate multiple parameters.

The IBM Case Manager REST protocol version identification (/v1) enables subsequent updates to the REST protocol.

The following example shows a URI for the particular task instance resource.

```
http://example.com:9080/CaseManager/CASEREST/v1/task
/7A75A997-0E42-406E-AZC4-EE55D7DER9PF?TargetObjectStore=MyExampleObjectStore
&Grouping=ROD
```

Special characters

Besides ASCII letters and decimal digits, you can use the following characters without any special notation in the case management REST URIs: $ - _ . + ! * ' ( ) ,

You must escape any other character, including spaces, double quotation marks (“ and ”), and percent signs (%). To escape a character in UTF-8 encoding, use `%xy` where xy is the two-digit hexadecimal value of the character. For example, %20 is the escaped representation of a space in a URI.

Symbolic names

When you design a case, Case Manager Builder assigns symbolic names for metadata objects such as property descriptions, document classes, and folder classes. You use the symbolic names when you refer to these metadata objects in calls to the methods that are provided by the IBM Case Manager REST protocol.

Typically, the symbolic name is generated from the display name that you enter for an object. The symbolic name begins with a letter and consists of uppercase and lowercase ASCII letters, decimal digits, and underscores. It can contain a maximum of 64 characters. The Content Platform Engine enforces that symbolic names are unique within an object store.

The IBM Case Manager REST protocol uses the symbolic name that is assigned by Content Platform Engine for the following items:
- Certain URI elements, including class names and object store names
- Values for query parameters that reference metadata objects
- The JSON payload for the names of object stores, case types, and activity types

Content Platform Engine requires symbolic names that are unique within the object store for metadata objects such as document classes and property templates. Instance objects, such as documents and folders, do not have symbolic names. For these objects, you use the GUID that is assigned to the object in the target object store.

In addition, solutions do not have symbolic names. To reference a solution, you use the solution name that is defined in Case Manager Builder.

Error responses

When a method call fails, the response code that the IBM Case Manager REST protocol returns indicates the type of error that occurred. For example, the response code `404 Not Found` indicates that the method did not find a resource such as the specified solution or case 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/1.1 500 Internal Server Error
Content-Type: application/json;charset-UTF-8
{
  "UserMessage":
  {
    "UniqueId":"FNRPA0024E",
    "Text":"FNRPA0024E IBM Case Manager Builder cannot connect to the Process
    Engine.",
    "Severity":"ERROR"
  }
  "UnderlyingDetails":
  {
    "Causes":
    [
      "Failed to connect to vworbbroker on hq-liquent:32776[100].  Check server
      connection.\nfilenet.pe.peorb.client.ORBServiceHelper$VWORBBrokerNotStarted:
      Failed to retrieve an IOR for vworbbroker. URL=http:\/\/hq-liquent:32776\
      /IOR\/FileNet.PE.vworbbroker.  Check PE server to make sure that vworbbroker
      process is started.",
      "Failed to retrieve an IOR for vworbbroker. URL=http:\/\/hq-liquent:32776\
      /IOR\/FileNet.PE.vworbbroker.  Check PE server to make sure that vworbbroker
      process is started."
    ]
  },
}
```

You can search for message information in the IBM Case Manager Information Center. Enter the value of the UniqueId element in the search field of the information center.

Common JSON payload for cases and case types

The IBM Case Manager REST protocol defines a JSON payload that is used in the methods that get or return information about a case or case type. This payload is also used by the external data service to obtain case information from an external data source.

The common JSON payload is used by methods for the following IBM Case Manager REST protocol resources:

Particular solution resource
The `GET` method uses the common payload to return a list of case types and the case type properties that are defined for a solution.

List of case types resource
The `GET` method uses the common payload to return a list of case types that are defined for a solution.

Particular case type resource

The GET method uses the common payload to return detailed property information for a case type.

The POST method uses the common payload to use the current property values to update information for dependent properties.

Cases resource

The POST method uses the common payload to create a case.

Particular case instance resource

The GET method uses the common payload to return information about a case.

The POST method uses the common payload to create a case by splitting an existing case.

The PUT method uses the common payload to update case information.

Payload parameters

The following code shows the structure of the full payload. However, not all methods use all parameters. See the specific method for the parameters that it uses in the payload.

```
{
"TargetObjectStore" : "<target object store name>",
"CaseType" : "<case type symbolic name>",
"CaseFolderId" : "<GUID of case folder>",
"DisplayName" : "<name displayed for  case type>",
"Description" : "<description of case type",
"CaseTitleProperty": "<property used as case title>",
"CaseIdentifier": "<case identifier>",
"ExternalDataIdentifier" : "<opaque data>",
"Properties":
[
{
   "SymbolicName"           : "<symbolic name>",
   "DisplayName"            : "<display name>",
   "Value"                  : <current property value>,
   "OriginalValue"          : <original property value>,
   "DisplayMode"            : "<readonly/readwrite>",
   "CustomValidationError"  : "<text of error>",
   "CustomInvalidItems"     : [<array of indexes>],
   "Description"            : "<property description>",
   "PropertyType"           : "<property type>",
   "Cardinality"            : "<single or multiple>",
   "Updatability"           : "<settability as defined in CE>",
   "Required"               : <required flag>,
   "Queryable"              : <true or false>,
   "Orderable"              : <true or false>,
   "Hidden"                 : <hidden flag>,
   "Inherited"              : <true or false>,
   "DefaultValue"           : <default property value>,
   "MaxValue"               : <maximum property value>,
   "MinValue"               : <minimum property value>,
   "MaxLength"              : <integer maximum length>,
   "HasDependentProperties" : <true or false>,
   "ChoiceList"             :
      {
      "DisplayName"         : "<display name for choic list>",
      "Choices"             :
            [
         {
            "ChoiceName"    : "<display name for choice>"
            "Value"         : <integer or string>
```

```
          },
        ]
      }
    }

    // ... additional properties

    ]
    "ClientContext":
    {
        "<key>":<value">,
        \\ additional key value pairs
    }
    }
```

The common JSON payload has the following attributes:

Table 18. Parameters for the common JSON payload

Parameter	Type	Description
TargetObjectStore	String	The symbolic name of the object store that contains the case. A symbolic name is called a unique identifier in IBM Case Manager.
CaseType	String	The symbolic name that is assigned to the case type.
ReturnUpdates	Boolean	A Boolean value that indicates whether the method is to return the property values after the case is created or updated. Set this parameter to true so that the method returns the case property values. By default, this parameter is set to false.
CaseFolderId	String	The GUID that identifies the root folder of an existing case.
DisplayName	String	The name that is displayed for the case in Case Manager Client.
Description	String	The description of the case.
CaseTitle Property	String	The name of the property that is used for the title of the case. By default, this parameter is set to `CmAcmCaseIdentifier`.
CaseIdentifier		The value of the `CmAcmCaseIdentifier` property for the case.

Table 18. Parameters for the common JSON payload (continued)

Parameter	Type	Description
ExternalData Identifier	String	A string that provides contextual information to indicate the state of the data that was returned by an external data service. The value of this parameter is set by an external data service. Typically, the service sets the parameter to reference the specific configurations that were used to define the attributes other than the property value. These attributes include settings for the minimum value, maximum value, choice list, and so on.Case Manager Client maintains the value or the parameter, but it does not change the value or parameter. The **ExternalDataIdentifier** parameter is required for the POST method of the particular case type resource. It is recommended that you include the **ExternalDataIdentifier** parameter in the payload whenever a method creates or updates a case. If you do not include the parameter, the IBM Case Manager REST protocol must establish another identifier internally.
Properties	Array	An array of JSON objects that represent the properties that have external data to merge with the underlying information. For a description of the attributes that can be included for each property, see Table 19.
ClientContext	JSON object	An object that contains a series of key value pairs that specify contextual information for a specific work item. This parameter is used to send information to an external data service when a case worker opens the work item.

Property attributes

You can include the following attributes for each property that is defined in the Properties parameter in the JSON payload.

Table 19. Property attributes in the common JSON payload

Attribute	Type	Description
SymbolicName	string	The symbolic name of the property.
DisplayName	string	The name that is displayed for the property in Case Manager Client.

Table 19. Property attributes in the common JSON payload (continued)

Attribute	Type	Description
Value	Boolean, datetime, float, integer, string, object	The value of the property. The value is returned in various response payloads based on the type of case the call is referencing, to indicate the current or working property value. • For a new case, the value starts out with a default value, which can be null. • For an existing case, the value starts out with the current value on the case. • An external data service can override this value. For a new case, the new value becomes the new working value before the case is created, and for an existing case it becomes the working value before the case is saved. • The user can also modify the working value. • The value can take various forms, depending on the type of property: – null – a string – a Boolean – an integer or a float – for a datetime type, a value in W3C format (for example, `2012-10-31T18:30:10`) – for an ID type, a string GUID – for a multivalued type, an array of the appropriate non-null type – for an object-valued type, another JSON object with information about the object that this property refers to
OriginalValue	Boolean, datetime, float, integer, string, object	The value currently persisted for the property in the repository. The IBM Case Manager REST protocol uses the **OriginalValue** parameter to determine whether the value that is specified by external data service is different from the value in the repository. • This parameter can be specified in certain input payloads and is preserved in the response payload, regardless of whether the property value is modified. • If not present in an input payload, it is not preserved in the response payload.

Table 19. Property attributes in the common JSON payload (continued)

Attribute	Type	Description
DisplayMode	string	A string that is returned by an external data service to specify whether Case Manager Client is to display the property value as read-only. An external data service can determine what the value of a property must be. If a value is predetermined, the field is rendered read-only from the user's perspective, but that value is saved when the case is saved or created. This mode is ignored if **Updatability** is not `readwrite` or `oncreate` for a new case. This mode has the following options: **readonly** The field is rendered as read-only, and the specified value is saved as the value of the property. **readwrite** The default setting. The field is be rendered writable, but **Updatability** takes preference.
Custom Validation Error	string	A message produced by an external data service, explaining why the existing value is invalid. An external data service can validate the current property values. It can leave an invalid value as-is and put a message in this field. • The presence of this attribute indicates that the value is considered invalid by an external data service. • The absence of this attribute indicates only that there is not an external data service that considers this property value to be invalid. The client can still consider the other attributes of the property when providing feedback to the user about the state of the property. An example use case is a customer ID property. An external data service can determine that the value of the property does not represent a valid customer ID without automatically modifying the value to a valid ID. In that case, the external data service can provide a custom validation error message.
Custom Invalid Items	array of indexes	A list of items on a multivalue list that are invalid, given as an array of indexes into the multivalue list of values. If an external data service validates a property value and the property is multivalue, it can also indicate the individual items of the multivalue list that are invalid. This attribute is applicable only if **CustomValidationError** indicates that the property is invalid. It does not need to be present even for a multivalue property. If **CustomValidationError** indicates that the property is invalid and this attribute is missing, then the entire property, rather than just individual items, should be considered invalid.
Description	string	The description of the property.
PropertyType	string	The data type of the property: • integer • float • boolean • string • datetime • id • object

Table 19. Property attributes in the common JSON payload (continued)

Attribute	Type	Description
`Cardinality`	string	One of the following values that indicates whether the property contains a single value or multiple values: • single • multi
`Updatability`	string	One of the following values that indicates whether a case worker can modify the property value: **readonly** Indicates that a case worker can read the property value but cannot modify the value. **readwrite** Indicates that a case worker can read and modify the property value. **oncreate** Indicates that a case worker can modify the property value only when creating a case.
`Required`	Boolean	A Boolean value that is set to true to indicate that a value is required for the property.
`Queryable`	Boolean	A Boolean value that is set to true if the property can be used in a query condition.
`Orderable`	Boolean	A Boolean value that is set to true if the property can be used in an `Order By` clause in a query condition.
`Hidden`	Boolean	A Boolean value that is set to true to indicate that the property is to be hidden in Case Manager Client.
`Inherited`	Boolean	A Boolean value that is set to true if this property is inherited from the superclass of the class.
`DefaultValue`	Boolean, datetime, float, integer, string, object	The default value that is specified for the property in Case Manager Builder. If no value is specified, this parameter is set to null.
`MaxValue`	datetime, float, integer	A number that indicates the maximum value of the property.
`MinValue`	datetime, float, integer	A number that indicates the minimum value of the property.
`MaxLength`	integer	A number that indicates the maximum length of the property value.

Table 19. Property attributes in the common JSON payload (continued)

Attribute	Type	Description

`ChoiceList`	object	A JSON object that contains array that defines a list of choices for the property value.
		
		The **`ChoiceList`** value can contain a flat list of choices:
		`"ChoiceList"  :`
		`{`
		`  "DisplayName"  : "<display name for choice list>",`
		`  "Choices"    :`
		`  [`
		`    {`
		`     "DisplayName" : "<display name for choice 1>`
		`     "Value"  : <value>`
		`    },`
		
		`    {`
		`      "DisplayName" : "<display name for choice 2>",`
		`      "Value"  : <value>`
		`    },`
		
		`    // More choices ...`
		`  ]`
		`}`

`HasDependent Properties`	Boolean	A Boolean value that is set to true by an external data service if other properties depend on the value of this property.

Currently, the only object-valued properties (OVPs) returned by this protocol are OVPs with single values. List and enum type OVPs are not supported. The protocol supports retrieving OVPs but does not support updating OVPs.

A non-null OVP value is represented in the JSON as in the following example:

```
"Value": {
  "Type": "reference",
  "ObjectStoreIdentity": "{DE6FC95A-3E90-42E2-9F3B-8B74C3945733}",
  "ClassIdentity": "{557F0B86-5C74-4F6D-BEA7-2B8C5476DBCF}",
  "ObjectIdentity": "{A9FC8EEC-FC7F-4B53-A5A0-73FC1E774FA7}"
},
```

Type Currently always "reference". Indicates that the other attributes of the JSON object define a reference to an object in the repository.

ObjectStoreIdentity
Indicates an identity for the object store that holds the object. Property values returned from the protocol always have an Object Store ID (GUID) as this value.

ClassIdentity
Identifies the class of the object. Property values returned from the protocol always have a Class ID (GUID) as this value.

ObjectIdentity
Identifies the object itself. Property values returned from the protocol always have an ID (GUID) as this value.

Related reference:

"Client context for work items" on page 107

"Symbolic names" on page 22

"GET method for the particular solution resource" on page 35

"GET method for the particular case type resource" on page 44

"POST method for the particular case type resource" on page 47
"POST method for the cases resource" on page 54
"GET method for the particular case instance resource" on page 57
"POST method for the particular case instance resource" on page 60
"PUT method for the particular case instance resource" on page 64

Getting information about deployed solutions

You can use the IBM Case Manager REST protocol to get information about deployed solutions. This information includes the object stores to which the solutions are deployed and the connection point that identifies the workflow stream, communications port, and isolated region number that is used by the solution.

"List of document classes resource"
"List of solutions resource" on page 33
"Particular solution resource" on page 35

List of document classes resource

The list of document classes resource provides a list of document classes that are defined in a solution. You can use this list to determine what classes of documents can be attached to cases and work items.

"GET method for the list of document classes resource"

GET method for the list of document classes resource:

The `GET` method for the list of document classes resource returns a list of the document classes that are defined in a solution. If the solution uses an IBM Content Manager repository, the method returns a list of the item types that are defined in the solution.

URI

/CASEREST/v1/solution/*{solution name}*/documenttypes

The URI for the `GET` method includes the following path element:

Table 20. Path element for the `GET` method

Name	Type	Required?	Description
`{solution name}`	String	No	The name of the solution for which the list of document classes is to be returned.

The URI for the `GET` method includes the following parameter:

Table 21. Parameter for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case type.

Request content

The request for this method contains no JSON content.

Response content

For each document class or item type, the method returns:

- The name of the document class or item type.
- The identifier of the item type. No identifier is returned for a document class.
- The name that is displayed for the document class or item type.
- The description of the document class or item type.
- A Boolean value that is set to true if the case worker has permission to create a document of this document class or item type. This value is always set to true for an item type.

The `GET` method also returns one of the following response codes:

Table 22. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully. The requested list of document classes was returned.
400 Bad Request	The required **`TargetObjectStore`** parameter was not specified or a parameter value was invalid.
404 Not Found	No document classes were found for the solution or the specified solution was not found. For information about the error, see the userMessage element in the JSON response.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests a list of all document classes that are defined for the Auto Claims solution:

```
#Request to get the document classes of a deployed solution
GET /CASEREST/v1/solution/Auto+Claims/documenttypes
?TargetObjectStore=MyTOS HTTP/1.1
Host: www.example.net
```

Example: `GET` method response for a Content Platform Engine object store

This sample code shows the list of all document classes that are defined for the Auto Claims solution in a Content Platform Engine object store:

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
    "DocumentTypes":
    [
        {
            "DocumentType": "AUTO_CollisionClaim",
            "DisplayName": "Collision Claim",
            "Description": "collision claim",
            "HasInstanceCreationRights": true
        },
        {
            "DocumentType": "Correspondence",
            "DisplayName": "Correspondence",
            "Description": "client correspondence",
            "HasInstanceCreationRights": true
        },
        ...
    ]
}
```

Example: `GET` method response for an IBM Content Manager repository

This sample code shows the list of all document classes that are defined for the Auto Claims solution in an IBM Content Manager repository.

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
    "DocumentTypes":
    [
        {
            "DocumentType": "Claim",
            "ItemTypeId": "10325", ?
            "DisplayName": "Liability Claim",
            "Description": "",
            "HasInstanceCreationRights": true
        },
        ...
    ]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

List of solutions resource

The list of solutions resource provides a list of all deployed solutions. This list can be useful to identify the servers to which you must redeploy an updated solution.

"GET method for the list of solutions resource"

Related information:

Solution List page

GET method for the list of solutions resource:

The `GET` method returns a list of the solutions that are deployed to all target object stores.

URI

/CASEREST/v1/solutions

Request content

The request for this method contains no JSON content.

Response content

For each solution, the method returns:

- Solution name
- Name of the target object store to which the solution is deployed
- Deployment status

- Connection point
- Web application ID

The `GET` method also returns one of the following response codes:

Table 23. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully. The requested list of solutions was returned.
404 Not Found	No solutions were found. For information about the error, see the userMessage element in the JSON response.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method response

This sample code requests a list of all solutions that are deployed to all target object stores.

```
GET http://example.com:9080/CaseManager/CASEREST/v1/solutionsHTTP/1.1
Host: www.example.net
```

Example: `GET` method response

This sample code shows the list that is returned in response to the request for a list of all solutions that are deployed to all target object stores,

```
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "Solutions":
  [
    {
      "SolutionName": "Automobile Claims",
      "SolutionFolderId": "{659C6566-4A6B-4328-A89A-27D2D08D0A1B}",
      "Description": "Solution for processing automobile claims",
      "TargetOS": "AutomobileClaimsOS",
      "Status": "Completed",
      "PEConnectionPoint": "PECP1",
      "WebAppID": "ABC"
    },
    {
      "SolutionName": "Fire Insurance Claims",
      "SolutionFolderId": "{18389232-FE4D-4400-8215-0FFA5A3F2C88}",
      "Description": "Solution for processing fire damage",
      "TargetOS": "FireInsuranceOS",
      "Status": "Failed",
      "PEConnectionPoint": "PECP1",
      "WebAppID": "8"
    }
  ]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

Particular solution resource

The particular solution resource provides information for a deployed solution. You can use this resource to get information about the case types that are defined for the solution.

"GET method for the particular solution resource"

GET method for the particular solution resource:

The `GET` method for the particular solution resource returns information about the case types that are defined for a solution.

URI

/CASEREST/v1/solution/*{SolutionName}*

The URI for the `GET` method includes the following path element:

Table 24. Path elements for the `GET` method

Name	Type	Description
{SolutionName}	String	The name of the solution for which information is to be returned.

The URI for the `GET` method includes the following parameter:

Table 25. Parameters for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the solution.

Request content

The request for this method contains no JSON content.

Response content

For each case type, the method returns a list of the properties and a detailed description of the views.

The `GET` method also returns one of the following response codes:

Table 26. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully. The response that is returned by the `GET` method includes the information for the specified solution.
400 Bad Request	One of the required parameters was missing or a parameter value was invalid.
404 Not Found	The solution that was specified in the request was not found. If a request is received for an object type that the external data service does not manage any data for, it must return status code 404: Not Found. The integration tier layer of IBM Case Manager treats this return status as if the particular object type did not have any external data associated with it. No error is returned to the IBM Case Manager caller.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests information about the case types that are defined for the Auto Loan solution:

```
GET /CASEREST/v1/solution/Auto+Loan+Solution
?TargetObjectStore=MyTargetObjectStore HTTP/1.1
Host: "www.example.net"
```

Example: `GET` method

This sample code shows the case type information that is returned for the Auto Loan solution:

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
"CaseTypes" : [
{
"CaseType" : "AUTO_CollisionClaimCase1",
"DisplayName" : "Collision Claim Case",
"Description" : "Case to process a collision claim",
"CaseTitleProperty": "CmAcmCaseIdentifier",
"Views":
{
  "CaseDataView":
  {
    "Fields":
    [
      { "FieldType": "property", "Name": "caseName" },
      { "FieldType": "property", "Name": "accountNumber" },
      { "FieldType": "group", "Label": "Home Address",
        "OpenState": false,
        "Fields":
        [
          {"FieldType": "property", "Name": "StreetAddressLine1"},
          {"FieldType": "property", "Name": "StreetAddressLine2"},
          {"FieldType": "property", "Name": "City"},
          {"FieldType": "property", "Name": "State"},
          {"FieldType": "property", "Name": "ZIPCode"},
        ]
      }
    ]
  },
  "CaseSummaryView":
  {
    "Fields":
    [
      { "FieldType": "property", "Name": "caseName" },
      { "FieldType": "property", "Name": "customerName" },
      { "FieldType": "property", "Name": "requestedLoanAmount" },
      { "FieldType": "property", "Name": "percentageDown" },
      { "FieldType": "property", "Name": "FicoScore" }
    ]
  },
  "CaseSearchView":
  {
    "Fields":
    [
      { "FieldType": "property", "Name": "caseName" },
      { "FieldType": "property", "Name": "customerName" },
      { "FieldType": "property", "Name": "accountNumber" },
      { "FieldType": "property", "Name": "requestedLoanAmount" }
    ]
}
}
},
{
"CaseType" : "AUTO_CollisionClaimCase2",
"CaseTiTleProperty": "CmAcmCaseIdentifier",
"Views": ...
}
],
"SolutionProperties":
[
{
      "SymbolicName": "AUTO_City",
      "DisplayName": "City",
      "Value": null,
      "DisplayMode": "readwrite",
      "Description": "City where home office is located",
      "PropertyType": "string",
      "Cardinality": "single",
      "Updatability": "readwrite",
      "Required": true,
      "Queryable": true,
      "Orderable": true,
      "Hidden": false,
      "Inherited": false,
      "DefaultValue": null,
      "MaxLength": 64,
      "ChoiceList": {
        "DisplayName": "CityChoiceList",
        "Choices": [
          {
            "ChoiceName": "Los Angeles",
            "Value": "Los Angeles"
          },
          {
            "ChoiceName": "San Diego",
            "Value": "San Diego"
          },
          {
            "ChoiceName": "San Francisco",
            "Value": "San Francisco"
          }
        ]
  }
}
]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"Common JSON payload for cases and case types" on page 23

Getting information about deployed case types

You can use the IBM Case Manager REST protocol to access information about the case types. The case types identify the kinds of cases that case workers can create with your application.

"List of case types resource" on page 38

"List of view definitions resource" on page 40

"List of discretionary task types resource" on page 42

"Particular case type resource" on page 44

"Case page resource" on page 51

List of case types resource

The list of case types resource provides a list of the case types that are available for a solution. You can use this list to populate a choice list from which a case worker can select the type of case to create. For example, the choice list might include case types such as Loan application or Manage dispute.

"GET method for the list of case types resource"

Related information:

Setting permissions for a case type class

GET method for the list of case types resource:

The `GET` method for the list of case types resource returns information about each case type that is defined for a specified solution.

URI

/CASEREST/v1/solution/*{solution name}*/casetypes

The URI for the `GET` method includes the following path element:

Table 27. Path element for the `GET` method

Name	Type	Description
{solution name}	String	The name of the solution for which the list of case types is to be returned.

The URI for the `GET` method includes the following parameter:

Table 28. Parameter for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case type.

Request content

The request for this method contains no JSON content.

Response content

For each case type, the method returns the following information:

DisplayName
The name that is displayed for the case type.

Description
The description of the case type.

HasInstanceCreationRights
A Boolean value that is set to `true` if the case worker has permission to create a case of this case type and a subfolder of the case type folder.

These permissions are set when you configure security by using IBM Administration Console for Content Platform Engine or by using IBM Case Manager administration client. Permission to create a case is set by selecting the **Create instance** right for the specific case type. Permission to create a subfolder is set by selecting the **Create subfolder** right for the case type folder.

HasAnnotationRights

A Boolean value that is set to `true` if the case worker has permission to add a case of this case type to a folder. A case worker must have this permission to split a case of this case type.

These permissions are set when you configure security by using IBM Administration Console for Content Platform Engine or by using IBM Case Manager administration client. Permission to create a case by splitting an existing case is set by selecting the **File in folder/Annotate** right for the case type folder.

The `GET` method also returns one of the following response codes:

Table 29. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully. The requested list of case types was returned.
400 Bad Request	The required **`TargetObjectStore`** parameter was not specified or a parameter value was invalid.
404 Not Found	Either the solution specified in the request URI was not found or no case types were found for the specified solution. For more information, see the userMessage element in the JSON response.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests a list of the case types that are defined for the deployed Auto Claims solution:

```
GET http://example.com:9080/CaseManager/CASEREST/v1/solution/Auto+Claims
/casetypes?TargetObjectStore=MyTOS HTTP/1.1
Host: www.example.net
```

Example: `GET` method

This sample code shows the list of case types that is returned for the deployed Auto Claims solution:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "CaseTypes":
  [
    {
      "CaseType": "AUTO_CollisionClaim",
      "DisplayName": "Collision Claim",
      "Description": "process a collision claim"
      "HasInstanceCreationRights": true,
      "HasAnnotationRights": true
    },
    {
      "CaseType": "AUTO_LiabilityClaim",
      "DisplayName" : "Liability Claim",
      "Description": "process a liability claim"
      "HasInstanceCreationRights": true,
      "HasAnnotationRights": true
    },
    ...
  ]
}
```

Related reference:
"Error responses" on page 23
"Case management REST resource URIs" on page 21
"Symbolic names" on page 22
"Common JSON payload for cases and case types" on page 23

List of view definitions resource

The list of view definitions resource represents the properties that are set in Case Manager Builder for the views defined for a case type.

"GET method for the list of view definitions resource"

GET method for the list of view definitions resource:

The `GET` method for the list of view definitions resource returns the properties for the Case Summary view, the Case Properties view, and the Case Search view.

URI

/CASEREST/v1/casetype/*{case type name}*/viewdefinitions

The URI for the `GET` method includes the following path element:

Table 30. Path element for the `GET` method

Name	Type	Description
{case type name}	String	The symbolic name of the case type for which view properties are to be returned.

The URI for the `GET` method includes the following parameters:

Table 31. Parameter for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the target object store that contains the case type.

Request content

The request for this method contains no JSON content.

Response content

For each view definition, the `GET` method returns a list of the properties that are displayed as fields in the view. To work with these properties, you can use IBM CMIS for FileNet Content Manager to obtain detailed information such as data types and lengths.

The `GET` method also returns one of the following response codes:

Table 32. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully and returned the requested view properties.
400 Bad Request	The **`TargetObjectStore`** parameter was not specified or the parameter value was invalid.

Table 32. Response codes for the `GET` *method (continued)*

Code	Description
404 Not Found	The case type that was specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests the view properties for the AUTO_FleetPurchase case type:

```
GET http://example.com:9080/CaseManager/CASEREST/v1/casetype
/AUTO_FleetPurchase/viewdefinitions?
TargetObjectStore=MyExampleObjectStore HTTP/1.1
Host: www.CaseMgmtExample.net
```

Example: `GET` method response

This sample code shows the view properties that are returned for the AUTO_FleetPurchase case type:

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "CaseTitleProperty": "CmAcmCaseIdentifier",
  "CaseDataView":
  {
    "Fields":
    [
      {"FieldType": "property", "Name": "caseName"},
      {"FieldType": "property", "Name": "accountNumber"},
      {"FieldType": "group", "Label": "Home Address",
         "OpenState": false,
         "Fields":
         [
           {"FieldType": "property", "Name": "StreetAddressLine1"},
           {"FieldType": "property", "Name": "StreetAddressLine2"},
           {"FieldType": "property", "Name": "City"},
           {"FieldType": "property", "Name": "State"},
           {"FieldType": "property", "Name": "ZIPCode"},
         ]
     }
    ]
 },
  "CaseSummaryView":
  {
    "Fields":
    [
      {"FieldType": "property", "Name": "caseName"},
      {"FieldType": "property", "Name": "customerName"},
      {"FieldType": "property", "Name": "requestedLoanAmount"},
      {"FieldType": "property", "Name": "percentageDown"},
      {"FieldType": "property", "Name": "FicoScore"}
    ]
  },
  "CaseSearchView":
  {
    "Fields":
    [
      {"FieldType": "property", "Name": "caseName"},
      {"FieldType": "property", "Name": "customerName"},
      {"FieldType": "property", "Name": "accountNumber"},
      {"FieldType": "property", "Name": "requestedLoanAmount"}
    ]
  }
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

List of discretionary task types resource

The list of discretionary task types resource provides a list of the user-created task types that are defined for a specified case type. You can use this list to display a choice list of the user-created tasks that a case worker can add to the case as needed.

"GET method for the list of discretionary task types resource"

GET method for the list of discretionary task types resource:

The `GET` method returns the properties for the user-created task types that are defined for a specified case type.

URI

/CASEREST/v1/casetype/{case type name}/discretionarytasktypes

The URI for the `GET` method includes the following path element:

Table 33. Path element for the `GET` method

Name	Type	Description
{case type name}	String	The symbolic name of the case type for which the list of user-created tasks is to be returned.

The URI for the `GET` method includes the following parameter:

Table 34. Parameter for the `GET` method

Name	Type	Required?	Description
TargetObjectStore	String	Yes	The symbolic name of the object store that contains the case type.

Request content

The request for this method contains no JSON content.

Response content

For each user-created task type, the method returns the following properties:

Table 35. Properties returned by the `GET` method

Property	Description
`Description`	The description that is defined for the task.
`HasInstanceCreationRights`	A Boolean value that is set to true if the current user can create an instance of this task type.

Table 35. Properties returned by the GET method (continued)

Property	Description
RequiresLaunchInfo	A Boolean value that is set to true if the GET method for the create new task resource must be called to obtain launch step information for the task. If this property is set to true, the **StepElement** property is required for the POST method for the create new task resource.
TaskClassId	The GUID for the Task class.
TaskDisplayName	The name of the task that is displayed in Case Manager Client.
TaskName	The symbolic name of the task.
IsHidden	A Boolean value that is set to true if the task is hidden from the user at run time.
IsContainer	A Boolean value that is set to true if the task is a container task.

The GET method also returns one of the following response codes:

Table 36. Response codes for the GET method

Code	Description
200 OK	The method completed successfully. The requested list of task types was returned.
400 Bad Request	The required **TargetObjectStore** parameter was not specified or a parameter value was invalid.
404 Not Found	The case type specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, refer to the userMessage element in the JSON response.

Example: GET method request

This sample code request a list of the user-created tasks for the Collision claim case type:

```
GET http://example.com:9080/CaseManager/CASEREST/v1/casetype
/AUTO_CollisionClaim/discretionarytasktypes
?TargetObjectStore=MyTOS HTTP/1.1
Host: www.example.net
```

Example: GET method response

This sample code shows the list of the user-created tasks that are returned for the Collision claim case type. The **TaskName** field that is returned specifies the symbolic name of the user-created task class. To create a user-created task, your application must first pass this symbolic name to the GET method for the create new task resource. Your application must then call the POST method for the create new task resource.

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
   "DiscretionaryTaskTypes":
   [
      {
            "TaskName": "AUTO_ContactCustomer",
            "TaskDisplayName": "Contact Customer",
            "Description": "phone, email or write to the customer",
            "TaskClassId": "{76DE6D7A-FC7D-4AD0-A109-DB9B9E63E7AE}",
            "HasInstanceCreationRights": true,
            "RequiresLaunchInfo": true,
        "IsHidden": false,
        "IsContainer": false
         },
         {
            "TaskName": "AUTO_ReadCollisionReport",
            "TaskDisplayName": "Read Collision Report",
            "Description": "read the collision report and police report",
            "TaskClassId": "{070AF241-C4FC-4E0A-86ED-BE017B68913F}",
            "HasInstanceCreationRights": true,
            "RequiresLaunchInfo":false,
        "IsHidden": false,
        "IsContainer": false
         }
         ...
   ]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"GET method for the create new task resource" on page 74

Particular case type resource

The particular case type resource provides information about the properties that are defined for a case type or a case. In preparation for creating a case, you can use this resource to get a list of the properties that are defined for the specific case type. For an existing case, you can use this resource to return updated information for dependent properties based on the current working values of the case properties.

"GET method for the particular case type resource"

"POST method for the particular case type resource" on page 47

Related reference:

"Request modes" on page 106

"Response to a request for case data" on page 108

"Client context for work items" on page 107

GET method for the particular case type resource:

The GET method for the particular case type resource returns the property information that you need to create a case of the specified case type.

If you are using an external data service, the GET method incorporates information from that service into property information that the method returns.

To create a case that reuses data from an existing case, you can specify the optional SourceCaseFolderId parameter to identify the source case. The IBM Case Manager REST protocol reuses the property values that are not null or empty from the source case for any matching properties in the new case.

URI

/CASEREST/v1/casetype/*{case type name}*

The URI for the `GET` method includes the following path element:

Table 37. Path element for the `GET` method

Name	Type	Description
{case type name}	String	The symbolic name of case type

The URI for the `GET` method includes the following parameters:

Table 38. Parameters for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case type. A symbolic name is called a unique identifier in IBM Case Manager.
`SourceCaseFolderId`	String	No	The GUID that identifies the root folder of an existing case from which data is to be reused in creating the case.

Request content

The request for this method contains no JSON content.

Response content

`GET` method returns the properties that are defined for the specified case type.

The `GET` method also returns one of the following response codes:

Table 39. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully. The response that is returned by the `GET` method includes the information for the specified case types.
400 Bad Request	The required **`TargetObjectStore`** parameter was missing or the parameter value was invalid.
404 Not Found	The case type specified in the request was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests information about the properties that are defined for the DH2_MyCase case type:

```
#Request to properties for case type DH2_MyCase
GET /CaseManager/CASEREST/v1/casetype/DH2_MyCase?TargetObjectStore=MyTOS
 HTTP/1.1
Host: www.example.net
```

Example: `GET` method

This sample code shows the information that is returned for the properties that are defined for the DH2_MyCase case type:

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "externalDataIdentifier": "-1,0",
  "properties": [
    {
      "symbolicName": "DH2_State",
      "required": true,
      "maxLength": 2,
      "hasDependentProperties": true,
      "choiceList": {
        "displayName": "StateChoiceList",
        "choices": [
          {
            "displayName": "New York",
            "value": "NY"
          },
          {
            "displayName": "California",
            "value": "CA"
          },
          {
            "displayName": "Nevada",
            "value": "NV"
          }
        ]
      }
    },
    {
      "symbolicName": "DH2_PropOne",
      "maxValue": 10,
      "minValue": 1,
      "hasDependentProperties": false
    },
    {
      "symbolicName": "DH2_MVInt",
      "value": [
        0,
        100
      ],
      "maxValue": 1000,
      "minValue": 0,
      "hasDependentProperties": true
    },
    {
      "symbolicName": "DH2_MVString",
      "required": true,
      "maxLength": 24,
      "hasDependentProperties": false,
      "choiceList": {
        "displayName": "MVStringChoiceList",
        "choices": [
          {
            "displayName": "One",
            "value": "One"
          },
          {
            "displayName": "Two",
            "value": "Two"
          },
          {
            "displayName": "Three",
            "value": "Three"
          },
          {
            "displayName": "Ten",
              "value": "Ten"
            },
            {
              "displayName": "Eleven",
              "value": "Eleven"
            },
            {
              "displayName": "Twelve",
              "value": "Twelve"
            }
          ]
        }
      },
      {
        "symbolicName": "DH2_City",
        "value": null,
        "displayMode": "readonly",
        "hidden": true,
        "required": true,
        "hasDependentProperties": false
      }
    ]
  }
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"Common JSON payload for cases and case types" on page 23

POST method for the particular case type resource:

The POST method is used to obtain the properties defined for a case type to create a case, optionally passing in the client context. This method is also used to obtain updated values for dependent properties as a case worker edits a case.

When you get the properties to create a case, call the POST method instead of the GET method if you need to pass contextual information to an external data service. The POST method includes the **clientContext** parameter that contains an array of key value pairs that specify the contextual information for a specific task.

An external data service can specify that a property has dependent properties. The values of the dependent properties are determined by the value of that property. You can call the POST method when the property value is modified so that it can return updated values for the dependent properties. For example, you might use an external data service to populate a choice list with cities from a state that a case worker selects. When a case worker selects California as the state, you call the POST method to populate the choice list with the appropriate California cities.

URI

/CASEREST/v1/casetype/*{case type name}*

The URI for the POST method includes the following path element:

Table 40. Path element for the POST method

Name	Type	Description
{case type name}	String	The symbolic name of case type

Request content for retrieving data for a new case

```
{
  "TargetObjectStore" : "<target object store name>",
  "RequestMode":"<request mode>",
  "ClientContext":
  {
    "<key>":"<value>",
    // More key value pairs"
  }
}
```

Request content for retrieving updates for dependent properties

```
{
  "TargetObjectStore" : "<target object store name>",
  "RequestMode":"<request mode>",
  "ExternalDataIdentifier":"<string set by the external data service>",
  "Properties",
  [
   {
     "symbolicName" : "<property name>",
     "value"        : "<current value>",
   }
   {
     // More properties
   }
  ],
  "ClientContext":
  {
    "<key>":"<value>",
    // More key value pairs
  }
}
```

Request parameters

Table 41. Request parameters for the `POST` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case type. A symbolic name is called a unique identifier in IBM Case Manager.
`RequestMode`	String	No	One of the following request modes that indicates the reason that the `POST` method is being called: **initialNewObject** Use this value if you are calling the `POST` method to get the properties for a new case. **inProgressChanges** Use this value if you are calling the `POST` method to update the values of dependent properties. The default value for the **`RequestMode`** is inProgressChanges.
`ExternalData Identifier`	String	Yes, if using an external data service to get values of dependent properties	A string that indicates the state of the data that was returned by the external data service. The value of this property is set by the service and is not modified by the client.

Table 41. Request parameters for the POST method (continued)

Name	Type	Required?	Description
Properties	Array	Yes	An array that contains values for the properties that are defined for the case type. For each property, you specify the symbolic name of the property and the value for the property. **Important:** The value must match the data type of the property.
ClientContext	JSON object	No	An array that contains a series of key value pairs that specify contextual information for a specific work item. This parameter is used to send information to an external data service when a case worker opens the work item.

Response content

The content of the response that is returned by the POST method depends on the setting of the **RequestMode** property. If this property is set to initialNewObject, the response contains all the properties that are defined for the specified case type. If the property is set to inProgressChanges, the response contains only those properties that were updated by an external data service based on a change to another property value.

The POST method also returns one of the following response codes:

Table 42. Response codes for the POST method

Code	Description
200 OK	The method completed successfully. The response that is returned by the POST method includes the information for the specified case types.
400 Bad Request	One of the required parameters was missing, or a parameter value was invalid.
404 Not Found	The case type that was specified in the request was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: POST method request

This sample code requests the updated values for the **DH2_City** property when a case worker selects *CA* for the **DH2_State** property:

```
POST /CaseManager/CASEREST/v1/casetype/DH2_MyCase
{
  "TargetObjectStore": "CMTOSDH",
  "ExternalDataIdentifier": "-1,0",
  "Properties": [

    // Properties not related to external data

    {
      "SymbolicName": "CmAcmCaseIdentifier",
      "Value": null
    },
    {
      "SymbolicName": "CmAcmCaseState",
      "Value": 0
    },

    // ...

    {
      "SymbolicName": "DH2_State",
      "Value": "CA"
    },
    {
      "SymbolicName": "DH2_PropOne",
      "Value": null
    },
    {
      "SymbolicName": "DH2_City",
      "Value": null
    },
    {
      "SymbolicName": "DH2_MVInt",
      "Value": [
        0,
        100
      ]
    },
    {
      "SymbolicName": "DH2_MVString",
      "Value": [ ]
    }
  ]
}
```

Example: `POST` method response

This sample code shows the choice list items that are returned when the **`DH2_City`** property is set to California:

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "externalDataIdentifier": "1,0",
  "properties": [
    {
      "symbolicName": "DH2_City",
      "hidden": false,
      "required": true,
      "hasDependentProperties": false,
      "choiceList": {
        "displayName": "CityChoiceList",
        "choices": [
          {
            "displayName": "Los Angeles",
            "value": "Los Angeles"
          },
          {
            "displayName": "San Diego",
            "value": "San Diego"
          },
          {
            "displayName": "San Francisco",
            "value": "San Francisco"
          }
        ]
      }
    }
  ]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"Common JSON payload for cases and case types" on page 23
"Request modes" on page 106
"Client context for work items" on page 107

Case page resource

The case page resource represents the physical identifier of the Case Details page, the Add Case page, or Split Case page that is used for a specific case type. You can use this identifier to open the page in the user interface.

"GET method for the case page resource"

GET method for the case page resource:

The `GET` method for the case page resource returns the page ID of the Case Details page, the Add Case page, or the Split Case page for a specific case type.

URI

/CASEREST/v1/casetype/*{case type name}*/page

The URI for the `GET` method includes the following path element:

Table 43. Path element for the `GET` method

Name	Type	Description
{case type name}	String	The symbolic name of the case type.

The URI for the `GET` method includes the following parameters:

Table 44. Parameters for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case type.
`PageType`	String	Yes	One of the following values that indicates the page for which the ID is to be returned: **`CaseCreationPage`** Returns the ID for the Add Case page. **`CasePage`** Returns the ID for the Case Details page. **`CaseSplitPage`** Returns the ID for the Split Case page.
`Role`	String	No	The name of the role for which the Case Details page ID is to be returned. Specify this parameter when the **`PageType`** parameter is set to `CasePage` to return the ID of the Case Details page that is used for a specific role. This parameter is not valid if the **`PageType`** parameter is set to `CaseCreationPage` or `CaseSplitPage`.

Request content

The request for this method contains no JSON content.

Response content

The method returns the page ID of the specified page type for the case type.

The `GET` method also returns one of the following response codes:

Table 45. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully and returned the requested page ID.
400 Bad Request	The required **`TargetObjectStore`** parameter or **`PageType`** parameter was not specified, or a parameter value was invalid.
404 Not Found	The case folder that was specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests the page ID for the Case Details page that is defined for the caseWorker role in the AUTO_CollisionClaim case type:

```
GET http://example.com:9080/CaseManager/CASEREST/v1/casetype
/AUTO_CollisionClaim/page?TargetObjectStore=MyTOS
&PageType=CasePage&Role=caseWorker HTTP/1.1
Host: www.example.net
```

Example: `GET` method response

This sample code shows the page ID for the Case Details page that is defined for the caseWorker role in the AUTO_CollisionClaim case type:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "PhysicalPageId": "e134f49999c112399a"
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"GET method for the list of view definitions resource" on page 40

Getting and changing case information

You can use the IBM Case Manager REST protocol to get and set case data, including case comments, case history, and tasks.

Cases and case folders

A case is represented as a case folder within the folder hierarchy in the target object store for a deployed solution. A case is filed in folder hierarchy under the `Cases` folder for its case type. A case folder contains the tasks, history, and comments that are associated with the case.

To prevent any one folder from being overloaded with too many objects, the new case folder is placed in a subfolder that is based on the year, month, and day that the case was created. The subfolder also includes a randomly generated number to identify the parent folder. The following example illustrates the general structure of the folder hierarchy for a deployed solution. In this example, the subfolder hierarchy is represented as follows:

- yyyy: The four-digit year
- mm: The two-digit month
- dd: The two-digit day
- pppp: The four-digit parent folder number

```
/IBM Case Manager
  /Solution Deployments
    /<solution name 1>
      /Case Types        /<Case Type 1a folder>
          /Cases
            /yyyy
              /mm
                /dd
                  /pppp
                    /<case folder. Name = sequence number>
                    ... (more case instance folders)
        /<Case Type 1b folder>
          /Cases
            /yyyy
              /mm
                /dd
                  /pppp
                    /<case folder. Name = sequence number>
                    ... (more case types for solution 1)
    /<solution name 2>
      /Case Types
        /<Case Type 2a folder>
          /Cases
            /yyyy
              /mm
                /dd
                  /pppp
                    /<case folder. Name = sequence number>
                    ...
        /<Case Type 2b folder>
          /Cases
            /yyyy
              /mm
                /dd
                  /pppp
                    /<case folder. Name = sequence number>
                    ... (more case types for solution 2)
    ... (more solutions)
```

"Cases resource"
"Particular case instance resource" on page 56
"Status of particular case resource" on page 67
"Related cases for a particular case resource" on page 68
"List of task instances resource" on page 71
"Create new task resource" on page 74
"Particular task instance resource" on page 80
"Case comments resource" on page 82
"Case history resource" on page 87

Cases resource

The cases resource represents the cases that are defined in your case management system. You can use this resource to create a case.

The cases resource creates a case that is represented as a case folder in the folder hierarchy in the target object store for a deployed solution. A case is filed in the folder hierarchy under the `cases` folder for its case type. To prevent any one folder from being overloaded with too many objects, the cases resource places the new case folder in a subfolder. The subfolder path is based on the year, month, and day that the case was created along with a unique number assigned to the parent folder.

"POST method for the cases resource"

Related reference:

"Client context for work items" on page 107

POST method for the cases resource:

The `POST` method for the cases resource creates a case by creating a case folder under the `cases` folder for its case type.

The property values that are submitted in the `POST` method request are validated by Content Platform Engine. If you use an external data service for the case type, the property values in the request are also validated by the IBM Case Manager REST protocol against the values that are returned by the service. The protocol validates the values against any property attributes that are set by the service, such as the minimum value, maximum value, and choice list.

When the case is saved, the value that was specified for a property in Case Manager Client is persisted for the case if the value meets the constraints that are set by the external data service. If a value is not specified for a property in Case Manager Client, the external data service can set a value that is persisted for the case.

URI

/CASEREST/v1/cases

Request content

```
{
  "CaseType": "<case type symbolic name>",
  "TargetObjectStore": "<target object store name>",
  "ReturnUpdates": <true or false>,
  "ExternalDataIdentifier" : "<string set by external data service">,
  "Properties" :
  [ // the array of case property values may be empty
    {
      "SymbolicName": "<symbolic name of case property>",
      "Value" : <property value>
    },
    ...
  ]
  "ClientContext":
  {
    "<key>":"<value>",
    // More key value pairs
  }
}}
```

Table 46. Request parameters for the `POST` method

Name	Type	Required?	Description
CaseType	String	Yes	The symbolic name that is assigned to the case type.

Table 46. Request parameters for the `POST` method (continued)

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case. A symbolic name is called a unique identifier in IBM Case Manager.
`ReturnUpdates`	Boolean	No	A Boolean value that indicates whether the method is to return the property values after the case is created. Set this parameter to `true` to force the method to return the case property values. By default, this parameter is set to `false`.
`ExternalData Identifier`	String	No	A string that indicates the state of the data that was returned by the external data service. **Tip:** Include this parameter in the request if a value was provided in response to a previous call to get data from the external data service.
`Properties`	Array	No	An array that contains values for the properties that are defined for the case type. For each property, you specify the symbolic name of the property and the value for the property. **Important:** The value that is specified for the property must match the data type of the property. You can use the particular case type resource to get a list of the properties that are defined for the case type.
`ClientContext`	Array	No	An array that contains a series of key value pairs that specify contextual information for a specific work item. This parameter is used to send information to an external data service when a case worker opens the work item.

Response content

The `POST` method returns the case title, case identifier, and case folder identifier for the new case folder. The `POST` method also returns one of the following response codes:

Table 47. Response codes for the `POST` method

Code	Description
201 Created	The method completed successfully. The `POST` method returns the identifier that is assigned to the new case folder.
400 Bad Request	One of the required parameters was missing, or a parameter value was invalid.
404 Not Found	The case type that was specified in the request was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `POST` method request

This sample code requests a case of the AUTO_CollisionClaim case type to be created and the property values to be returned after the case is created.

Tip: If the **`ReturnUpdates`** parameter is set to true, the response from the `POST` method is similar to the response from the `GET` method.

```
POST http://example.com:9080/CaseManager/CASEREST/v1/cases
HTTP/1.1
Host: www.CaseMgmtExample.net
Content-Type: charset.json;charset-UTF-8
{
    "CaseType": "AUTO_CollisionClaim",
    "TargetObjectStore": "ATOSME",
    "ReturnUpdates": false,
    "Properties":
    [
      {
        "SymbolicName"  : "AUTO_ClaimDate",
        "Value"         : "2010-07-16T21:50:36Z",
      },
      {
        "SymbolicName"  : "AUTO_ClaimStatus",
        "Value"         : "0",
      }
    ]
}
```

Example: `POST` method response

This sample code shows the property values that are returned for the new AUTO_CollisionClaim case:

```
HTTP 1.1 201 OK Created
Content-Type: application/json;charset-UTF-8
{
  "CaseTitle": "DH2_MyCase_000000100402",
  "CaseIdentifier": "DH2_MyCase_000000100402",
  "CaseFolderId": "{A42BE8EB-848F-4CBD-B2F7-64FAF2CE7081}"
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"Common JSON payload for cases and case types" on page 23

"Client context for work items" on page 107

Particular case instance resource

The particular case instance resource represents a case. You can use this resource to return or update the property values for a case.

You can also use the `POST` method of this resource to split an existing case to form two cases. For example, an insurance claim for a car accident might initially be filed as a single case. After further investigation, you might decide to split the original case into two cases. The original case tracks the claim for damage to the car, and the second case covers the claim for injuries.

"GET method for the particular case instance resource" on page 57

"POST method for the particular case instance resource" on page 60

"PUT method for the particular case instance resource" on page 64

Related reference:

"Response to a request for case data" on page 108

"Client context for work items" on page 107

GET method for the particular case instance resource:

The GET method for the particular case instance resource returns the properties that are defined for a case.

URI

/CASEREST/v1/case/*{case folder id}*

The URI for the GET method includes the following path element:

Table 48. Path element for the GET method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case for which the method is to return properties.

The URI for the GET method includes the following parameter:

Table 49. Parameter for the GET method

Name	Type	Required?	Description
TargetObjectStore	String	Yes	The symbolic name of the object store that contains the case.

Request content

The request for this method contains no JSON content.

Response content

The GET method returns the following information for a specified case folder:
- The symbolic name of the case that is represented by the case folder
- A list of the case properties and their current values

If you are using an external data service to populate the case properties, the GET method includes information from the external service in the response. Typically, the values provided by the external data service are the same as the current property values for the case. However, if the values for a property are different, the **Value** attribute for the property differs from the **OriginalValue** attribute.

The GET method also returns one of the following response codes:

Table 50. Response codes for the GET method

Code	Description
200 OK	The method completed successfully, and the list of properties for the specified case was returned.
400 Bad Request	A required parameter was missing, or the parameter value was invalid.
404 Not Found	The specified case folder was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: **GET** method request

This sample code requests the current property values for case C5AB1E9D-30D1-4D21-ADDF-F248FF1354B7:

```
GET
http://localhost:9081/CaseManager/CASEREST/v1/case/
C5AB1E9D-30D1-4D21-ADDF-F248FF1354B7
?TargetObjectStore=CMTOSDH
```

Example: **GET** method response

This sample code shows the current property values that are returned for case C5AB1E9D-30D1-4D21-ADDF-F248FF1354B7:

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "Properties": [

    {
      "SymbolicName": "DateCreated",
      "DisplayName": "Date Created",
      "Value": "2011-04-28T18:49:55Z",
      "OriginalValue": "2011-04-28T18:49:55Z",
      "DisplayMode": "readwrite",
      "Description": "The date and time this object was created.",
      "PropertyType": "datetime",
      "Cardinality": "single",
      "Updatability": "readonly",
      "Required": false,
      "Queryable": true,
      "Orderable": true,
      "Hidden": false,
      "Inherited": true,
      "DefaultValue": null,
      "MaxValue": "9999-12-31T23:59:59Z",
      "MinValue": "1753-01-01T00:00:00Z",
      "HasDependentProperties": false
    },

    // Additional properties omitted

    {
      "SymbolicName": "CmAcmCaseIdentifier",
      "DisplayName": "Case Identifier",
      "Value": "DH2_MyCase_000000100602",
      "OriginalValue": "DH2_MyCase_000000100602",
      "DisplayMode": "readwrite",
      "Description": "A specially formatted identifier for
        Case Folder instances, consists of Case Folder subclass
        symbolic class name, \"_\" and then a  12 digit sequence
        number with leading zeros.",
      "PropertyType": "string",
      "Cardinality": "single",
      "Updatability": "readwrite",
      "Required": false,
      "Queryable": true,
      "Orderable": true,
      "Hidden": false,
      "Inherited": true,
      "DefaultValue": null,
      "MaxLength": 85,
      "HasDependentProperties": false
    },
    {
```

```
    "SymbolicName": "CmAcmCaseState",
    "DisplayName": "Case State",
    "Value": 2,
    "OriginalValue": 2,
    "DisplayMode": "readwrite",
    "Description": "An integer choice property that defines the possible
      the possible states of Case Folder instance.",
    "PropertyType": "integer",
    "Cardinality": "single",
    "Updatability": "readwrite",
    "Required": true,
    "Queryable": true,
    "Orderable": true,
    "Hidden": false,
    "Inherited": true,
    "DefaultValue": 0,
    "MaxValue": null,
    "MinValue": null,
    "ChoiceList": {
      "DisplayName": "CmAcmCaseStateChoiceList",
      "Choices": [
       {
        "ChoiceName": "New",
        "Value": 0
       },
       {
        "ChoiceName": "Initializing",
        "Value": 1
       },
       {
        "ChoiceName": "Working",
        "Value": 2
       },
       {
        "ChoiceName": "Complete",
        "Value": 3
       },
       {
        "ChoiceName": "Failed",
        "Value": 4
       }
      ]
    },
    "HasDependentProperties": false
  },
  {
    "SymbolicName": "DH2_State",
    "DisplayName": "State",
    "Value": "CA",
    "OriginalValue": "CA",
    "DisplayMode": "readwrite",
    "Description": "State where home office is located",
    "PropertyType": "string",
    "Cardinality": "single",
    "Updatability": "readwrite",
    "Required": true,
    "Queryable": true,
    "Orderable": true,
    "Hidden": false,
    "Inherited": false,
    "DefaultValue": null,
    "MaxLength": 2,
    "ChoiceList": {
      "DisplayName": "StateChoiceList",
      "Choices": [
      {
        "ChoiceName": "New York",
          "Value": "NY"
        },
        {
          "ChoiceName": "California",
          "Value": "CA"
        },
        {
          "ChoiceName": "Nevada",
          "Value": "NV"
        }
        ]
      },
      "HasDependentProperties": true
    },

    // Additional properties omitted


  ]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"Common JSON payload for cases and case types" on page 23

POST method for the particular case instance resource:

The POST method for the particular case instance resource returns information for a specific case. You can call this method to create a new case from an existing case. You can also call this method to return information for a case. If you use an external data service, you can pass client context information in the request to provide contextual information about work items.

URI

/CASEREST/v1/case/*{case folder id}*

The URI for the POST method includes the following path element:

Table 51. Path element for the POST method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case that is to be split.

Request content for creating a split case

```
{
    "CaseType": "case type symbolic name",
    "TargetObjectStore": "target object store name",
    "OperationDescription": "operation description,
    "Operation": "split",
    "ExternalDataIdentifier": "string set by the external data service",
    "Properties" :
    [ // the array of case property values may be empty
        {
        "SymbolicName": "symbolic name of case property",
        "Value" : "property value"
        },
        ...
    ],
    "DocumentFiling":
    [ // the array of folders to have documents filed in
        {
        "FolderName": "path to case subfolder, or just '/'",
        "DocumentId": "vsid for P8 document or PID for CM8 document"
        },
        ...
    ]
}
```

No data from the original case is used when creating the new split case. You must pass in the property values you want to set on the new split case by using the properties attribute.

Request content for getting case data

```
{
    "TargetObjectStore": "<target object store name>",
    "Operation": "fetchProperties",
    "ClientContext":
    {
        "key": "value",
        // More key value pairs
    }
}
```

Request content for adding a case relationship

```
{
    "TargetObjectStore": "target object store name",
    "Operation": "relate",
    "CaseFolderId": "GUID of target case",
    "OperationDescription": "operation description - this parameter is optional",
    "RelationshipCategory": "category name - this parameter is optional",
    "TwoWayRelationship": "true/false defaults to true - this parameter is optional"
}
```

Tip: A user can create relationships between cases even if that user does not have write permission for the case folders. The request to relate a case can succeed even if the user has only read permission on the folder.

Request content for removing a case relationship

```
{
    "TargetObjectStore": "target object store name",
    "Operation": "unrelate",
    "RelationshipId": "GUID of relationship to delete",
    "OperationDescription": "operation description - this parameter is optional"
}
```

Tip: A user can remove relationships between cases even if the user does not have write permission on the case folders. The request to remove the relationship between cases can succeed even if the user has only read permission on the folders.

Request parameters

Table 52. Request parameters for the `POST` method

Name	Type	Required?	Description
`CaseType`	String	Yes	The symbolic name that is assigned to the case type of the case that is to be created by the `POST` method. The **`CaseType`** parameter is not required for the relate operation, and it is not required for the unrelate operation.
`TargetObjectStore`	String	Yes	The symbolic name of the object store that is to contain the new case. A symbolic name is called a unique identifier in IBM Case Manager.
`Operation Description`	String	No	Text that describes the action that is indicated by the `POST` method.
`Operation`	String	Yes	One of the following operations that the `POST` method is to run: **`split`** Specify this option to reuse data from the current case to create a case. **`fetchProperties`** Specify this option to return data for the current case. **`relate`** Specify this option to relate another case to the current case. **`unrelate`** Specify this option to remove the relationship between this case and a related case.
`ExternalData Identifier`	String	No	A string that indicates the state of the data that was returned by the external data service. The external identifier is set by the external data service when the properties are fetched for the first time. This identifier is passed back to the external data service for splitting a case or creating a case. The value of this property is set by the service and is not modified by the client. **Tip:** If you are using an external data service and the **`Operation`** parameter is set to `split`, you can include the **`ExternalDataIdentifier`** parameter in the request, since the identifier was set when the properties were fetched. This parameter is not required if the **`Operation`** parameter is set to `fetchProperties`, because the identifier might not be set by the external data service.
`Properties`	Array	No	An array that contains values for the properties that are defined for the case type. For each property, you specify the symbolic name of the property and the value for the property. **Important:** The value must match the data type of the property. You can use the particular case type resource to get a list of the properties that are defined for the case type.
`DocumentFiling`	Array	No	An array that identifies any documents to be attached to the new case and the folder in which the documents are to be filed. Use the version ID to identify a document. Use a slash (/) to indicate the root folder. Use a slash and the folder name to indicate a subfolder under the root folder, for example, `/folder1`.
`ClientContext`	Array	No	An array that contains a series of key value pairs that specify contextual information for a specific work item. This parameter is used to send information to an external data service when a case worker opens the work item.

Table 52. Request parameters for the `POST` method (continued)

Name	Type	Required?	Description
`RelationshipId`	String	No	The GUID of the related case you want to remove the relationship from.
`RelationshipCategory`	String	No	An optional string that describes the category of the relationship.
`TwoWayRelationship`	String	No	Indicates whether the related case also has a relationship with the current case. The value of the parameter must be true or false. The default value is true.

Response content

The content of the response that is returned by the `POST` method depends on the operation that you are running. If you are running the `split` operation, the method returns the identifier of the new case that was created by reusing data from an existing case. If you are running the `fetchProperties` operation, the method returns the case properties.

The `POST` method also returns one of the following response codes:

Table 53. Response codes for the `POST` method

Code	Description
200 OK	The method completed successfully, and the case was created. The response that is returned by the `POST` method contains the case folder ID for the new case.
201 OK	The method completed successfully, and the case relationship was created. The response that is returned by the `POST` method contains the ID of the new case relationship.
400 Bad Request	A required parameter was missing, or a parameter value was invalid.
404 Not Found	The case folder that was specified in the request was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

`POST` method request for splitting a case

This sample code shows a request for the case with the ID ending in EE55D8BCF2ED to be split to create a case of the `My_casetype` case type:

```
#Request
POST /CASEREST/v1/case/9E45A997-0E42-406E-AAC4-EE55D8BCF2ED
 HTTP/1.1
Host: www.CaseMgmtExample.net
Content-Type: application/json;charset-UTF-8
{
  "CaseType": "My_casetype",
  "TargetObjectStore": "myTargetOS",
  "OperationDescription": "splitting case1 to case2",
  "Operation": "split"
  "Properties" :
  [
    {
      "SymbolicName"  : "MY_property1",
      "Value" : "property1Value"
    }
  ],
  "DocumentFiling" :
  [ // the array of folders to have documents filed in
    {
      "FolderName": "/CaseSubFolder1",
      "DocumentId": "12345678-0000-0000-0000-aabbccddeeff"
    }
  ]
}
```

POST method response for splitting a case

This sample code shows the case folder ID that is returned for the new case that was created by splitting the case given in the POST request:

```
HTTP 1.1 201 OK Created
Content-Type: application/json;charset-UTF-8
{
    "CaseFolderId": "{12345678-1234-1234-1234-aabbccddeeff}"
}
```

POST method request for relating a case

This sample code shows a request for a case to be related to a case of the My_casetype case type:

```
#Request
POST /CASEREST/v1/case/9E45A997-0E42-406E-AAC4-EE55D8BCF2ED
 HTTP/1.1
Host: www.CaseMgmtExample.net
Content-Type: application/json;charset-UTF-8
{
  "CaseType": "My_casetype",
  "TargetObjectStore": "myTargetOS",
  "Operation": "relate",
  "CaseFolderId": "{12345678-1234-1234-1234-aabbccddeeff}",
  "OperationDescription": "description of operation",
  "RelationshipCategory": "category name",
  "TwoWayRelationship": "true"
}
```

POST method response for create operation

This sample code shows the relationship ID that is returned for the relationship that was created:

```
HTTP 1.1 201 OK Created
Content-Type: application/json;charset-UTF-8
{
    "RelationshipId": "{12345678-1234-1234-1234-aabbccddeeff}"
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"Common JSON payload for cases and case types" on page 23

"Client context for work items" on page 107

PUT method for the particular case instance resource:

The PUT method for the particular case instance resource updates the case properties in the specified case folder with new values. Optionally, the method returns the full list of case properties with the updated values.

The property values that are submitted in the PUT method request are validated by Content Platform Engine. If you use an external data service for the case type, the property values in the request are also validated by the service. The service validates the values against any property attributes that are set by the service, which include the minimum value, maximum value, and choice list.

When the case is saved, the value that was specified for a property in Case Manager Client is persisted for the case if the value meets the constraints set by the external data service. If a value is not specified for a property in Case Manager Client, the external data service can set a value that is persisted for the case.

URI

/CASEREST/v1/case/*{case folder id}*

The URI for the `PUT` method includes the following path element:

Table 54. Path element for the `PUT` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case for which the `PUT` method is to update property values.

Request content

```
{
    "TargetObjectStore": "<target object store name>",
    "ExternalDataIdentifier" : "<string set by external data service>",
    "ReturnUpdates": <true or false>
    "Properties" :
    [ // the array of case property values can be empty
        {
        "SymbolicName": "<symbolic name of case property>",
        "Value" : <property value>
        },
        ...
    ]
    "ClientContext":
    {
      "<key>":"<value>",
      // More key value pairs
    }
}
```

Table 55. Request parameters for the `PUT` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that is to contain the new case. A symbolic name is called a unique identifier in IBM Case Manager.
`ExternalData Identifier`	String	No	A string that indicates the state of the data that was returned by the external data service. **Tip:** Include this parameter in the request if a value was provided in response to a previous call to get data from the external data service.
`ReturnUpdates`	Boolean	No	A Boolean value that indicates whether the method is to return the updated case property values. Set this parameter to `true` to force the method to return the case property values. By default, this parameter is set to `false`.

Table 55. Request parameters for the `PUT` method (continued)

Name	Type	Required?	Description
`Properties`	Array	No	An array that contains values for the properties that are defined for the case type. For each property, you specify the symbolic name of the property and the value for the property. **Important:** The value that is specified for the property must match the data type of the property. You can use the particular case type resource to get a list of the properties that are defined for the case type.
`ClientContext`	Array	No	An array that contains a series of key value pairs that specify contextual information for a specific task.

Response content

By default, the `PUT` method returns one of the following response codes. Optionally, the method also returns the full list of case properties with the updated values.

Table 56. Response codes for the `PUT` method

Code	Description
200 OK	The method completed successfully. The case was updated with the new property values.
400 Bad Request	The required **`TargetObjectStore`** parameter was missing, or the parameter value was invalid.
404 Not Found	The specified case folder was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Examples: `PUT` method request

This sample code requests the update of values for the five properties of a case (with ID ending with 354B7):

```
PUT
http://localhost:9081/CaseManager/CASEREST/v1/case/
C5AB1E9D-30D1-4D21-ADDF-F248FF1354B7

{
  "TargetObjectStore":"CMTOSSH",

  "Properties":
  [
    {"SymbolicName":"DH2_State","Value":"NV"},
    {"SymbolicName":"DH2_PropOne","Value":8},
    {"SymbolicName":"DH2_City","Value":"Reno"},
    {"SymbolicName":"DH2_MVInt","Value":[0,101,300,340]},
    {"SymbolicName":"DH2_MVString","Value":["One","Three","Sixty"]}
  ]
}
```

Examples: `PUT` method response

This sample code shows the response code that is returned after the property values are updated in case C5AB1E9D-30D1-4D21-ADDF-F248FF1354B7. Because the **`ReturnUpdates`** parameter was not set in the request, the method does not return the updated property values.

```
HTTP 1.1 200 OK
Content-Type: application/json;charset-UTF-8
{ }
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"Common JSON payload for cases and case types" on page 23

"Client context for work items" on page 107

Status of particular case resource

The status of particular case resource represents status information about a case. You can use this resource to determine whether a case completed successfully.

"GET method for the status of particular case resource"

GET method for the status of particular case resource:

The `GET` method for the status of particular case resource returns a value that indicates the status of a specified case. The status indicates the state of the case as complete, failed, initializing, new, or working.

URI

/CASEREST/v1/case/*{case folder id}*/status

The URI for the `GET` method includes the following path element:

Table 57. Path element for the `POST` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case for which status is to be returned.

The URI for the `GET` method includes the following parameter:

Table 58. Parameter for the `GET` method

Name	Type	Required?	Description
TargetObjectStore	String	Yes	The symbolic name of the object store that contains the case type. A symbolic name is called a unique identifier in IBM Case Manager.

Request content

The request for this method contains no JSON content.

Response content

The `GET` method returns the case identifier, the date that the case was created, and the date that the case was last modified. In addition, the method returns one of the following values that indicates the status of the case. You can query this value to determine whether a case was successfully created.

Table 59. Case status values

Value	Description
Complete	All tasks that are associated with the case are completed.
Failed	The case was not created. The response might still include a case ID and a case creation date if the case folder was created.
Initializing	The case is being created, but is not yet ready to be worked on.
New	The process of creating the case started.
Working	The case was created and is ready to be worked on.

The `GET` method also returns one of the following response codes:

Table 60. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully. The response that is returned by the `GET` method includes the status of the specified case.
400 Bad Request	The required **`TargetObjectStore`** parameter was missing, or the parameter value was invalid.
404 Not Found	The case specified in the request was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests the status of the case in a specified case folder:

```
#Request
GET /CASEREST/v1/case/9E45A997-0E42-406E-AAC4-EE55D8BCF2ED/status
?TargetObjectStore=MyExampleObjectStore
 HTTP/1.1
Host: www.CaseMgmtExample.net
```

Example: `GET` method

This sample code shows the response to the request, with the status of the case:

```
#Response
  HTTP/1.1 200 OK
  Content-Type: application/json;charset-UTF-8
{
    "Status": "Working",
    "CaseIdentifier":"MY_Case_000000100105",
    "DateCreated":" 2010-07-16T21:50:36Z",
    "DateLastModified":"2010-07-16T21:50:36Z"
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

Related cases for a particular case resource

The related cases for a particular case resource represent the set of cases that are related to a specific case. You can use this resource to return a list of the related cases. For example, you can return a list of the cases that were created by splitting the current case.

IBM Case Manager supports the following relationships between cases:

Table 61. Case relationships

Relationship	Description
split source	The related case was created when the current case was split.
split target	The current case was created when the related case was split.

"GET method for the related cases for a particular case resource"

GET method for the related cases for a particular case resource:

The `GET` method for the related cases for a particular case resource returns information for each case that is related to a specified case. Related cases include the case that was split to create the current case and any cases that were created by splitting the current case. Results can be filtered by type or category of relationship.

URI

/CASEREST/v1/case/*{case folder id}*/cases

The URI for the `GET` method includes the following path element:

Table 62. Path element for the `GET` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case for which related cases are to be returned.

The URI for the `GET` method includes the following parameters:

Table 63. Parameter for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case. A symbolic name is called a unique identifier in IBM Case Manager.
`RelationshipType`	String	No	The type of relationship between the case that is returned and the case that is initiating the request. Use this parameter to filter the results by the type of relationship.
`RelationshipCategory`	String	No	The category of the relationship between the case that is returned and the case that is initiating the request. Use this parameter to filter the results by the category of relationship. Use this parameter only if RelationType is "Related".

Request content

The request for this method contains no JSON content.

Response content

For each case that is related to the specified case, the `GET` method returns the following properties:
- Status
- Case title

- Case identifier
- Date created
- Creator
- Relationship type
- Relationship ID
- Relationship category

The `GET` method also returns one of the following response codes:

Table 64. Response codes for the `GET` method

Code	Description
201 Created	The method completed successfully and returned the requested case comments.
400 Bad Request	The required **TargetObjectStore** parameter was not specified, or the parameter value was invalid.
404 Not Found	The case folder that was specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests a list of the cases that are related to a specified case (with ID ending in F2ED):

```
#Request
GET /CASEREST/v1/case/9E45A997-0E42-406E-AAC4-EE55D8BCF2ED/cases
?TargetObjectStore=MyExampleObjectStore
 HTTP/1.1
Host: www.CaseMgmtExample.net
```

Example: `GET` method response

This sample code shows the response to the request, with the list of the cases that are related to the case in the request:

```
#Response
   HTTP/1.1 200 OK
   Content-Type: application/json;charset-UTF-8
[
{
    "Status": "Working",
    "CaseTitle": "MY_Case_000000100105",
    "CaseIdentifier":"MY_Case_000000100105",
    "CaseFolderId":"1D56A997-0E42-406E-AAC4-EE55D8BCF2ED",
    "DateCreated":" 2010-07-16T21:50:36Z",
    "Creator": "Admin",
    "RelationshipType": "split source",
    "RelationshipId": "{12345678-1234-1234-1234-aabbccddeeff}"
},
{
    "Status": "Working",
    "CaseTitle": "MyCaseTitle",
    "CaseIdentifier":"MY_Case_000000100106",
    "CaseFolderId":"2E67A997-0E42-406E-AAC4-EE55D8BCF2ED",
    "DateCreated":" 2010-07-16T21:50:36Z",
    "Creator": "Admin",
    "RelationshipType": "split target",
    "RelationshipID": "{22345678-1234-1234-1234-aabbccddeeff"}
},
{
        "Status": "Working",
        "CaseTitle": "MY_Case_000000100107",
        "CaseIdentifier":"MY_Case_000000100107",
        "CaseFolderId":"3F47B997-0E42-406E-AAC4-EE55D8BCF2ED",
        "DateCreated":" 2010-07-16T21:50:36Z",
        "Creator": "Admin",
        "RelationshipType": "split target"
        "RelationshipID": "{32345678-1234-1234-1234-aabbccddeeff"}
},
{
        "Status": "Working",
        "CaseTitle": "MY_Case_000000100107",
        "CaseIdentifier": "MY_Case_000000100107",
        "CaseFolderId": "3F47B997-0E42-406E-AAC4-EE55D8BCF2ED",
        "DateCreated": "2010-07-16T21:50:36Z",
        "Creator": "Admin",
        "RelationshipType": "Related",
        "RelationshipId": "{42345678-1234-1234-1234-aabbccddeeff}",
        "RelationshipCategory": "user profile"
}]
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

List of task instances resource

The list of task instances resource provides a list of all the tasks instances that are running for a particular case instance.

"GET method for the list of task instances resource"

GET method for the list of task instances resource:

The `GET` method returns a collection that lists all of the tasks that are running for a particular case. In the collection, the tasks are grouped according to whether they are required, optional, or disabled.

URI

/CASEREST/v1/case/*{case folder id}*/tasks

The URI for the `GET` method includes the following path element:

Table 65. Path element for the `GET` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case for which tasks are to be returned.

The URI for the `GET` method includes the following parameters:

Table 66. Parameters for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case. A symbolic name is called a unique identifier in IBM Case Manager.

Table 66. Parameters for the `GET` method (continued)

Name	Type	Required?	Description
`Grouping`	String	Yes	The identifier that indicates grouping for the tasks. You must set this parameter to `ROD`, which represents the following groups: **Required** This group includes tasks for which the `RequiredState` property is set to REQUIRED_BY_USER or REQUIRED_BY_INCLUSIVE. **Optional** This group includes tasks that are enabled and for which the `RequiredState` property is set to OPTIONAL. **Disabled** This group includes tasks that are disabled and for which the `DisabledState` property is set to DISABLED_BY_USER, DISABLED_BY_EXCLUSIVE, or DISABLED_BY_ABORTED. The `GET` method does not return groups that are empty. The groups can be returned in any order. Within each group, the tasks are ordered first by the task state and then by the task name.

Request content

The request for this method contains no JSON content.

Response content

For the task, the method returns:
- The required state of the task
- The disabled state of the task
- The launch mode state of the task
- The date the task was created
- The task identifier
- The task name
- The task number
- The date the task was last modified
- Whether the task is hidden
- Whether the task is a container
- The process instance ID
- The date the task was last restarted
- The restart count
- The roster name

The `GET` method also returns one of the following response codes:

Table 67. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully and returned the requested list of tasks.
400 Bad Request	The required **`TargetObjectStore`** parameter or **`Grouping`** parameter was not specified, or a parameter value was invalid.

Table 67. Response codes for the `GET` method (continued)

Code	Description
404 Not Found	The case folder that was specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests a list of all the tasks that are running for a specified case (with ID ending in F2ED):

```
GET http://example.com:9080/CaseManager/CASEREST/v1/case
/9E45A997-0E42-406E-AAC4-EE55D8BCF2ED/tasks
?TargetObjectStore=MyExampleObjectStore&Grouping=ROD
HTTP/1.1
Host: www.CaseMgmtExample.net
```

Example: `GET` method response

This sample code shows the response to the request, with the list of tasks that are running for the case given in the request. If a task is in the failed state, the response also includes a **`FailureReason`** element that describes the reason for the failure. The text provided for this element might not be available in languages other than English.

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "Optional":
  [
    {
      "RequiredState": 0, "TaskState": 3, "DisabledState": 0,
      "LaunchMode": 0, "DateCreated": "2010-07-16T21:50:36Z",
      "TaskId": "{3B5C8E64-43FE-4188-AC72-457A4B8E374C}",
      "TaskName": "ETECase2 Task number 2",
      "DateLastModified": "2010-07-16T21:50:36Z",
      "IsHidden": false,
      "IsContainer": false,
      "ProcessInstanceId": "0907E35E7DC03B4FA03F6B6767633FB2",
      "LastRestartDate": "2010-07-16T21:50:36Z",
      "RestartCount": "1",
      "RosterName": "MySolution1"
    }
  ],
  "Required":
  [
    {
      "RequiredState": 1, "TaskState": 1, "DisabledState": 0,
      "LaunchMode": 4, "DateCreated":"2010-07-16T21:50:36Z",
      "TaskId": "{CB3F1916-8D03-44C8-9598-23589D9ED78F}",
      "TaskName": "ETECase2 Task number 1",
      "DateLastModified": "2010-07-16T21:50:36Z"
      "IsHidden": false,
      "IsContainer": false,
      "ProcessInstanceId": "0907E35E7DC03B4FA03F6B6767633FB1",
      "LastRestartDate": null,
      "RestartCount": "0",
      "RosterName": "MySolution1"
    }
  ]
}
```

Related reference:
"Error responses" on page 23
"Case management REST resource URIs" on page 21
"Symbolic names" on page 22

Create new task resource

By using the create new task resource, a case worker can add a user-created task to a case. You can use the `GET` method that is defined for this resource to retrieve the launch step information for the selected user-created task type. You can then use the `POST` method to add a new user-created task of that type to the case.

You use this resource only to create a user-created task. Tasks that are not user-created are created automatically either when a case is created or, for repeatable tasks, as needed.

"`GET` method for the create new task resource"
"`POST` method for the create new task resource" on page 77

`GET` method for the create new task resource:

The `GET` method returns the launch step information for the specified task type that is required to add a user-created task to the case. The launch step information is passed to the `POST` method to start the user-created task.

Remember: You must call the `GET` method if the **`RequiresLaunchInfo`** property in payload returned by the `GET` method for the list of discretionary task types resource is set to true.

URI

/CASEREST/v1/case/*{case folder id}*/tasktype/*{symbolic task name}*

The URI for the `GET` method includes the following path elements:

Table 68. Path elements for the `GET` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case to which the task is to be added.
{symbolic task name}	String	The symbolic name of the task type to be used for the new task.

The URI for the `GET` method includes the following parameter:

Table 69. Parameter for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case.

Request content

The request for this method contains no JSON content.

Response content

The `GET` method returns the following information that is required to add a user-created task to the case:

- Attachments
- System properties
- Workflow groups
- Data fields
- Step processor

The `GET` method also returns one of the following response codes:

Table 70. Response codes for the `GET` method

Code	Description
200 OK	The method completed successfully. The requested task type information was returned.
400 Bad Request	The required **`TargetObjectStore`** parameter was not specified or a parameter value was invalid.
404 Not Found	The case folder ID or the symbolic task name specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `GET` method request

This sample code requests the launch step information for an `AUTO_ContactCustomer` task:

```
GET http://example.com:9080/CaseManager/CASEREST/v1/case
/12345678-abcd-dcba-4321-12345678/tasktype/
AUTO_ContactCustomer?TargetObjectStore=MyTOS HTTP/1.1
Host: www.example.net
```

Example: `GET` method response

This sample code shows the launch step information for an `AUTO_ContactCustomer` task:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "attachments": {},
  "systemProperties":
  {
    "responses": ["yes", "no"],
    "mapName": "Workflow",
    "stepId": 0,
    "stepName": "LaunchStep",
    "caseFolderId": "{8CA37883-9BA1-4513-AF94-120EA4255A2B}",
    "workflowName": "ETE_ETECase3_ETECase3Task1",
    "selectedResponse": "",
    "workObjectNumber": "D931E58C31E1DE44BDF519E88565614F",
    "subject": "ETE_ETECase3_ETECase3Task1",
    "authoredMapName": "Workflow",
    "instruction": ""
  },
  "workflowGroups":
  {
    "F_Trackers":
    {
      "value" : [],
      "desc": "",
      "mode": 3,
      "modified": false,
      "type": 64,
      "name": "F_Trackers",
      "isArray": true
    }
  },
  "dataFields":
  {
    "ETEProperty1":
    {
      "value": true,
      "desc": "",
      "mode": 3,
      "modified": false,
      "type": 4,
      "name": "ETEProperty1",
      "isArray": false
    },
    "ETEProperty2":
    {
      "value": 163,
      "desc": "",
      "mode": 3,
      "modified": false,
      "type": 1,
      "name": "ETEProperty2",
      "isArray":false
    },
    "ETEProperty3":
    {
      "value": "TestStringChoice1",
      "desc": "",
      "mode": 3,
      "modified": false,
      "type": 2,
      "name": "ETEProperty3",
      "isArray": false
    },
    "ETEProperty4":
    {
      "value": 3.1415926535,
      "desc": "",
      "mode": 3,
      "modified": false,
      "type": 8,
      "name": "ETEProperty4",
      "isArray": false
    },
    "ETEProperty5":
    {
      "value": "2010-07-05T19:21:24Z",
      "desc": "",
      "mode": 3,
      "modified": false,
      "type": 16,
      "name": "ETEProperty5",
      "isArray":false
    }
  },
  "stepProcessor":
  {
    "width":800,
    "height":600,
    "applicationName":"",
    "appType":32,
    "id":455,
    "name":"ETE_LaunchPage",
    "processorType":4,
    "locations":{"8":"123456"}
  }
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

"GET method for the list of discretionary task types resource" on page 42

`POST` method for the create new task resource:

The `POST` method adds a new user-created task to a case by passing in the workflow launch step information that was returned by the preceding `GET` method.

URI

/CASEREST/v1/case/*{case folder id}*/tasks

The URI for the `POST` method includes the following path element:

Table 71. Path element for the `POST` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case to which the task is to be added.

Request content

The `POST` method can create the user-created task successfully, but the response might be lost in transit. In that situation, you can implement logic to send the request again. The request returns one of the following responses:

- A 201 Created response code is returned if the initial `POST` request was never received by the server, but the second request was received and successfully processed.
- A 200 OK response code is returned if the initial `POST` request was received by the server and successfully processed, but the response was lost. In this situation, the second request is treated as a duplicate `POST` request.
- Any other return code indicates that an error occurred.

```
Content-Type: charset.json;charset-UTF-8
{   "TaskTypeName": "<Symbolic task type name>",
    "TaskName": "<Task name to create>",
    "StepElement": <JSON object returned by previous GET method>
}
```

Table 72. Request parameters for the `POST` method

Name	Type	Required?	Description
`TaskTypeName`	String	Yes	The symbolic name of the task type.
`TaskName`	String	Yes	The name of the task that is being created.

Table 72. Request parameters for the POST method (continued)

Name	Type	Required?	Description
StepElement	Object	No	A JSON object that contains the information that is required to launch the task. The **StepElement** parameter is required if the **RequiresLaunchInfo** property that is returned by the GET method for the list of discretionary task types resource is set to true.

Response content

The POST method returns the identifier for the new task. The POST method also returns one of the following response codes:

Table 73. Response codes for the POST method

Code	Description
201 Created	The method completed successfully. The workflow that is associated with the task is started.
200 OK	A duplicate POST request was detected, so the method does not create a task or start the workflow. However, the response that is returned is the same as the response that is returned for the initial request.
400 Bad Request	The required **TargetObjectStore** parameter was missing, or a parameter value was invalid.
404 Not Found	The case folder specified in the request was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: POST method request

This sample code submits a request to create a new user-created task.

```
POST http://example.com:9080/CaseManager/CASEREST/v1/case
/9E45A997-0E42-406E-AAC4-EE55D8BCF2ED/tasks?
TargetObjectStore=MyExampleObjectStore HTTP/1.1
Host: www.CaseMgmtExample.net
Content-Type: charset.json;charset-UTF-8
{
  "TaskTypeName": "AUTO_TakeCustomerToLunch",
  "TaskName": "Take customer to lunch",
  "StepElement":
  {
    "attachments": {},
    "systemProperties":
    {
      "responses": ["yes","no"],
      "mapName": "Workflow",
      "stepId": 0,
      "stepName": "LaunchStep",
      "caseFolderId": "{8CA37883-9BA1-4513-AF94-120EA4255A2B}",
      "workflowName": "ETE_ETECase3_ETECase3Task1",
      "selectedResponse": "yes",
      "workObjectNumber": "D931E58C31E1DE44BDF519E88565614F",
      "subject": "ETE_ETECase3_ETECase3Task1",
      "authoredMapName": "Workflow",
      "instruction": ""
    },
    "workflowGroups":
    {
      "F_Trackers":
      {
```

```
    "value": [],
    "desc": "",
    "mode": 3,
    "modified": false,
    "type": 64,
    "name": "F_Trackers",
    "isArray": true
  }
},
"dataFields":
{
  "ETEProperty1":
  {
    "value": true,
    "desc": "",
    "mode": 3,
    "modified": false,
    "type": 4,
    "name": "ETEProperty1",
    "isArray": false
  },
  "ETEProperty2":
  {
    "value": 163,
    "desc": "",
    "mode": 3,
    "modified": false,
    "type": 1,
    "name": "ETEProperty2",
    "isArray": false
  },
  "ETEProperty3":
  {
    "value": "TestStringChoice1",
    "desc": "",
    "mode": 3,
    "modified": false,
    "type": 2,
    "name": "ETEProperty3",
    "isArray": false
  },
  "ETEProperty4":
  {
    "value": 3.1415926535,
    "desc": "",
    "mode": 3,
    "modified": false,
    "type": 8,
    "name": "ETEProperty4",
    "isArray": false
  },
  "ETEProperty5":
  {
    "value": "2010-07-05T19:21:24Z",
    "desc":"",
    "mode": 3,
    "modified": false,
    "type": 16,
    "name": "ETEProperty5",
    "isArray": false
  }
},
"stepProcessor":
{
  "width":800,
  "height":600,
  "applicationName":"",
      "appType":32,
      "id":455,
      "name":"ETE_LaunchPage",
      "processorType":4,
      "locations":{"8":"123456"}
    }
  }
}
```

Example: `POST` method response

This sample code shows the response that is returned when the new user-created task is created.

```
HTTP/1.1 201 Created
Location: http://www.CaseMgmtExample.net/task/{task ID}
Content-Location: http://www.CaseMgmtExample.net/task/{task ID}
Content-Type: application/json;charset-UTF-8
{
  "TaskId": "{CB3F1916-8D03-44C8-9598-23589D9ED78F}"
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

Particular task instance resource

The particular task instance resource represents an instance of a task. You can use this resource to change the state of a task, for example, from disabled to started.

"`PUT` method for the particular task instance resource"

`PUT` method for the particular task instance resource:

The `PUT` method updates a specified task. Typically, you use this method to start, enable, or disable a task.

To avoid unnecessary Content Platform Engine errors, the `PUT` method ensures that the task is in the correct state before the task is updated. If a case worker requests to disable a task, the `PUT` method ensures that the task is not in a working or complete state. If the task is in a working or complete state, the method ignores the request. If a case worker requests to start a task, the `PUT` method ensures that the task is in a ready state.

Because multiple case workers might decide a task must be enabled, the `PUT` method does not return an error when a request is made to enable a task that is already enabled. Instead, the method always returns an updated version of the task.

URI

/CASEREST/v1/task/*{taskId}*

The URI for the `PUT` method includes the following path element and parameters:

Path element

Table 74. Path element for the `PUT` method

Name	Type	Description
{taskId}	String	The GUID for the task instance that is to be updated.

Parameters

Table 75. Parameters for the `PUT` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the task. A symbolic name is called a unique identifier in IBM Case Manager.
`Grouping`	String	Yes	The identifier that indicates grouping for the tasks. You must set this parameter to `ROD`, which represents the following groups: **Required** This group includes tasks for which the `RequiredState` property is set to `REQUIRED_BY_USER` or `REQUIRED_BY_INCLUSIVE`. **Optional** This group includes tasks that are enabled and for which the `RequiredState` property is set to `OPTIONAL`. **Disabled** This group includes tasks that are disabled and for which the `DisabledState` property is set to `DISABLED_BY_USER`, `DISABLED_BY_EXCLUSIVE`, or `DISABLED_BY_ABORTED`. The `PUT` method does not return groups that are empty.

Request content

```
{
  "action": "<start or enable or disable or stop or restart>"
}
```

Important: For a restart request to succeed, the user who makes the call must have create rights for the roster of the solution. For a stop request to succeed, the user who makes the call must have read rights on all queues in the solution. By default, all users have read rights to queues, but if you have customized Content Platform Engine security, you must add read rights for users who might call this API.

Response content

For the task that is updated, the method returns:
- The required state of the task
- The disabled state of the task
- The launch mode state of the task
- The date the task was created
- The task identifier
- The task name
- The task number
- The date the task was last modified

The `PUT` method also returns one of the following response codes:

Table 76. Response codes for the `PUT` *method*

Code	Description
200 OK	The method completed successfully. No content is returned.
400 Bad Request	The **TargetObjectStore** parameter or the **Grouping** parameter was not specified, or a parameter value was invalid.
404 Not Found	The task specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: `PUT` method

This example disables a task.

Request example

```
PUT http://example.com:9080/CaseManager/CASEREST/v1/task
/7A75A997-0E42-406E-AZC4-EE55D7DER9PF?TargetObjectStore=MyExampleObjectStore
&Grouping=ROD HTTP 1.1
Host: www.example.net
{
  "Action": "disable"
}
```

Response example

If a task is in the failed state, the response also includes a **FailureReason** field that describes the reason for the failure.

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "Required":
  [
    {
      "RequiredState": 1,
      "TaskState": 1,
      "DisabledState": 0,
      "LaunchMode": 4,
      "DateCreated": "2010-07-16T21:50:36Z",
      "TaskId": "{CB3F1916-8D03-44C8-9598-23589D9ED78F}",
      "TaskName": "ETECase2 Task number 1",
      "DateLastModified": "2010-07-16T21:50:36Z"
    }
  ]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

Case comments resource

The case comments resource represents the comments that are associated with a specific case. A case comment can be associated with the case or with a document, task, or work item in the case. The format of the comment varies depending on the component with which it is associated. You can use this resource to retrieve comments for a case and to add a comment to a case.

Restriction: You cannot use the case comments resource to update or delete case comments.

"`GET` method for the case comments resource"

"`POST` method for the case comments resource" on page 85

`GET` method for the case comments resource:

The `GET` method returns a collection that contains all comments of a specified type for a case. The comments are returned in reverse chronological order based on the creation date.

URI

/CASEREST/v1/case/*{case folder id}*/comments

The URI for the `GET` method includes the following path element:

Table 77. Path element for the `GET` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case for which comments are to be returned.

The URI for the `GET` method includes the following parameters:

Table 78. Parameters for the `GET` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case.
`CommentType`	String	Yes	One of the following values to indicate the type of comments to be returned: • `Task` • `Case` • `Document` • `WorkItem`
`ItemId`	String	No	The identifier that indicates the specific document or task or work item for which comments are to be returned. For a document, specify the version series ID. For a task or a work item, specify the GUID for the task. You must specify this parameter if you set the **`CommentType`** parameter to `Task`, `Document`, or `WorkItem`. Do not specify this parameter if you set the **`CommentType`** parameter to `Case`.
`WorkflowNumber`	String	No	The workflow number that indicates the specific work item for which comments are to be returned. This parameter is optional if the **`CommentType`** parameter is set to `WorkItem`. If you do not specify this parameter, the `GET` method returns all work item comments for the matching task. Work items are always associated with a task. **Tip:** After a work item completes, it no longer exists. Access to the work item is not possible. However, if you have the workflow number for the work item, you can still retrieve the individual comments for that work item.

Request content

The request for this method contains no JSON content.

Response content

For each comment, the GET method returns the following information:
- The name of the person who created the comment
- The date the comment was created
- The comment text
- The comment context that indicates the action, such as adding a case or a document, that was being taken when the comment was created

The GET method also returns one of the following response codes:

Table 79. Response codes for the GET method

Code	Description
201 Created	The method completed successfully and returned the requested case comments.
400 Bad Request	The required **TargetObjectStore** parameter or **CommentType** parameter was not specified, or a parameter value was invalid.
404 Not Found	The case folder that was specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: GET method request

This sample code requests all the comments for a case:

```
GET http://example.com:9080/CaseManager/CASEREST/v1/case
/9E45A997-0E42-406E-AAC4-EE55D8BCF2ED/comments?
TargetObjectStore=MyExampleObjectStore&CommentType=Case HTTP 1.1
Host: www.example.net
```

Example: GET method

This sample code shows all the comments for a case:

```
{
  "Comments":
  [
    {
      "Id": "{5E42A997-0F47-446E-AFC4-EE55D8BCF5PP}",
      "Creator": "Bob",
      "CommentContext": 101,
      "DateCreated": "2010-04-07T14:30Z",
      "CommentText": "New request from Bob at GimmeCars.com"
    },
    {
      "Id": "{9E45A997-0E42-406E-AAC4-EE55D8BCF2EA}",
      "Creator": "Mary",
      "CommentContext": 102,
      "DateCreated": "2010-04-07T15:30Z",
      "CommentText": "Fast-track Bob's request – very good customer"
    }
  ]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

`POST` method for the case comments resource:

The `POST` method adds a comment to a case, a task, a document, or a work item that is associated with a specific case folder.

URI

/CASEREST/v1/case/*{case folder id}*/comments

The URI for the `POST` method includes the following path element:

Table 80. Path element for the `POST` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case to which the comment is to be added.

The URI for the `POST` method includes the following parameter:

Table 81. Parameter for the `POST` method

Name	Type	Required?	Description
`TargetObjectStore`	String	Yes	The symbolic name of the object store that contains the case.

Request content

The content of a request to create a comment depends on the type of comment that you are creating:

Table 82. Request elements required to create different comment types

Request element	Comment types	Description
CommentType	Case, document, task, work item	One of the following values that indicates the type of comments to be returned: • `Task` • `Case` • `Document` • `WorkItem`
CommentContext	Case, document, task, work item	A number that indicates the action, such as adding a document or case, that was being taken when this comment was created. This value is based on the choice list that is defined in the `CmAcmActionChoiceList` object in the target object store.
CommentText	Case, document, task, work item	A string that contains the text for the comment.
ItemId	Document, task, work item	The identifier that indicates the specific document, task, or work item for which the comment is to be added. For a document, specify the version series ID. For a task or a work item, specify the GUID for the task.
DocumentTitle	Document	The title that is assigned to the document in Content Platform Engine.
WorkClassName	Work item	The name of the work class that describes the attributes of the work item, such as data fields, a security configuration, and event logging options. In most cases, a work class corresponds to a workflow roster.
StepName	Work item	The name of the step that contains the work item.
WorkflowNumber	Work item	The work object number that indicates the specific work item for which comments are to be returned.

Table 82. Request elements required to create different comment types (continued)

Request element	Comment types	Description
Response	Work item	The response that was used to process the work item.

Response content

For each comment that is added, the method returns the following information:
- The comment context, which indicates the action that was being taken when this comment was created
- The date the comment was created
- The comment identifier
- The text of the comment
- The creator of the comment

The `POST` method also returns one of the following response codes:

Table 83. Response codes for the `POST` method

Code	Description
200 OK	The method completed successfully. The new comment was added to the case. The response header includes the URI for the comment.
400 Bad Request	One of the required parameters was not specified, or a parameter value was invalid. For information about the error, see the userMessage element in the JSON response that was returned by this method.
404 Not Found	The case folder that was specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response that was returned by this method.

Example: `POST` method request

This sample code requests that a comment be created for a document in a case:

```
POST http://example.com:9080/CaseManager/CASEREST/v1/case/
9E45A997-0E42-406E-AAC4-EE55D8BCF2ED/comments?
TargetObjectStore=MyExampleObjectStore HTTP 1.1
Host: www.example.net
{
  "CommentType" : "Document",
  "CommentContext" : 402,
  "CommentText" : "this is a sample comment for a document in a case",
  "ItemId" : "B9BA42F3-CD30-4C93-BE8B-BDE0BC85AA4F",
  "DocumentTitle" : "Sample Document for My Case"
}
```

Example: `POST` method response

This sample code shows the comment that was created for a document in a case. The **`Id`** value in the response specifies the identifier for the comment that was added.

```
201 Created
{
  "CommentContext":402,
  "DateCreated":"2010-07-21T23:15:40Z",
  "Id":"{C1D63E6A-0CEC-433E-A6B9-C0EA0FDEFB53}",
  "CommentText":"This is a sample comment for a document in a case",
  "Creator":"P8Admin"
}
```

Example: JSON payload for a task comment

```
{
  "CommentType" : "Task",
  "CommentContext" : 202,
  "CommentText" : "This is a sample comment for a task in a case",
  "ItemId" : "B4DD9C04-46B4-4295-8EA0-1C0DB95C6C74"
}
```

Example: JSON payload for a work item comment

```
{
  "CommentType" : "WorkItem",
  "CommentContext" : 301,
  "CommentText" : "This is a sample comment for a work item in a case",
  "ItemId" : "B4DD9C04-46B4-4295-8EA0-1C0DB95C6C74",
  "WorkClassName" : "_work_class_name",
  "StepName" : "test_step_name",
  "Response" : "test_response",
  "WorkflowNumber" : "78FE3D3856F047408B29ECA140EE90B7"
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

Case history resource

The case history resource represents the history for a specific case. You can use this resource to retrieve the entries that make up the case history. The case history shows information such as creation dates, comments, and such, about the case.

The case history that is maintained by IBM Case Manager is based on the Content Platform Engine audit feature. The entries are stored as event objects in the Event table that is in the database for the object store. The case history entries are audit events that are configured so that they can be retrieved by using the GUID of the case folder. Therefore, not all audit entries correspond to a case history entry.

Tip: You can enable and configure auditing through the Content Engine API or in the IBM Case Manager administration client. If the Content Platform Engine audit feature is not configured correctly, the information available in case histories might not be what you expect.

"GET method for the case history resource"

Related information:

Auditing concepts

Configuring auditing

GET method for the case history resource:

The GET method returns the entries that make up the case history. You can set parameters to return only entries for specific object types, such as folders or tasks, or for specific event types, such as creation or deletion of objects. If you do not set any parameters, all entries are returned. In addition, you can specify whether you want the complete information for the entries that are returned or only the summary information.

To make it easier to display the entries, the `GET` method always returns a string value for the **PropertyValue** element, regardless of the type of the property. Therefore, you do not convert integer, float, or datetime properties for display.

URI

/CASEREST/v1/case/*{case folder id}*/history

The URI for the `GET` method includes the following path element:

Table 84. Path element for the `GET` method

Name	Type	Description
{case folder id}	String	The GUID that identifies the root folder of the case for which history is to be returned.

The URI for the `GET` method includes the following parameters. You can combine the object types, such as documents and folders, and event types, such as creation, to obtain specific information. For example, you can return a summary of the additions of documents and folders by specifying the following parameters in the `GET` method:

```
&ObjectTypes=Document+CmAcmCaseFolder+CmAcmCaseSubfolder&EventTypes=Creation
```

Table 85. Parameters for the `GET` method

Name	Type	Required?	Description
TargetObjectStore	String	Yes	The symbolic name of the object store that contains the case.
BatchSize	Integer	No	The maximum number of entries to be returned. If you do not set this parameter, the method returns a maximum of 200 audit entries. **Tip:** For best results, set the **BatchSize** parameter to no more than 200.
ContinuationToken	String	No	The value of the Continuation element that is returned in the JSON response for the previous call to the `GET` method. Omit this parameter from the request to retrieve the first batch of entries. Specify this parameter in the next request to retrieve the next batch of entries. Enter the value as follows (without quotation marks): `ContinuationToken=34832908d930ddkdj390di3kj` If the Continuation element that is returned in the JSON response contains a null string, there are no more entries to be returned.

Table 85. Parameters for the GET method (continued)

Name	Type	Required?	Description
ObjectTypes	String	No	The object types for which entries are to be returned. Enter one or more object types by using spaces to separate multiple types. The following list identifies the symbolic names of Content Engine objects that you might need: **CmAcmCaseComment** Return entries for comments on the case. **CmAcmCaseFolder** Return entries for case folders. **CmAcmCaseSubfolder** Return entries for case subfolders. **CmAcmCaseTask** Return entries for tasks. **CmAcmVersionSeriesComment** Return entries for comments on a document. **CmAcmWorkItemComment** Return entries for comments on work items. **Document** Return entries for documents that are associated with the case. When appropriate, the `GET` method returns entries for subclasses of the selected object types. For example, the `GET` method returns entries for subclasses of the `Document` object type automatically. You do not specify each subclass by name. If you do not specify the **ObjectTypes** parameter, the method returns entries for all object types.
EventTypes	String	No	The type of event for which entries are to be returned. Enter one or more event types by using spaces to separate multiple types. You can query any subclass of the Content Engine `ObjectChangeEvent` class. If you do not specify the **EventTypes** parameter, the method returns entries for all event types.
AdditionalProperties	String	No	A list of the symbolic names of the properties to include in the "AdditionalProperties" JSON element in the returned payload. Enter one or more property names. Use spaces to separate multiple names.
AdditionalFilter	String	No	The property expression to be included in the `WHERE` clause of the case history query. This expression is a UTF-8 encoded URL, and must comply with Content Engine SQL syntax.

Request content

The request for this method contains no JSON content.

Response content

The `GET` method returns the entries in the case history based on the specified objects and events. The `GET` method also returns one of the following response codes:

Table 86. Response codes for the GET method

Code	Description
200 OK	The method completed successfully and returned the requested entries.
400 Bad Request	The required **TargetObjectStore** parameter was not specified, or a parameter value was invalid.
404 Not Found	The case folder that was specified in the request URI was not found.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Retrieving batches of case history entries

The first time that you call the GET method, you can set the **BatchSize** parameter to specify the number of entries. However, you do not set the **ContinuationToken** parameter. The GET method returns the specified number of entries in reverse chronological order, beginning with the newest entry in the log. If there are more entries in the case history, the method also returns a continuation token. You must include the continuation token in the query parameters to retrieve the next batch of entries.

To retrieve the next batch of entries, you make a second call to GET method by using the same values for all parameters except the **ContinuationToken** parameter. You set the **ContinuationToken** parameter to the continuation token that is returned by the preceding method call. The method then returns the next set of entries.

Important: For the GET method to return the entries correctly, all the query parameters, except for the continuation token, must be the same in subsequent calls.

You can continue to call the GET method by using the continuation token to return all entries for the case. When there are no more entries, the method does not return a continuation token.

To allow a case worker to view the results that were returned in a previous call to the GET method, you must maintain a string array that contains the nonnull continuation tokens. You can then use the array values to retrieve a specific batch of entries.

For example, assume that you store the continuation tokens in an array named *A* and that this array contains *X* tokens. *X* is also the index of the array element in which the next continuation token are stored. The initial value of *X* is 0 because there are no continuation results in the array *A*.

When the GET method returns the first batch of entries, the continuation token in the response is saved in *A[0]* and the value of *X* increases by one to 1 because it starts from 0. When the GET method returns the second batch of entries, the continuation token in the response is saved in *A[1]* and the value of *X* increases by one to 2.

After the third batch of entries is returned, *X* is equal to 3 and the continuation token from the fourth call to the GET method will be stored in *A[2]*. To retrieve the previously returned batches again, you set the **ContinuationToken** parameter to the following values:

- A[1] to retrieve the third batch of entries

- A[0] to retrieve the second batch of entries
- Null to retrieve the first batch of entries

The general rules for determining the continuation token are:
- If X-1 >= 0, then A[X-1] is the continuation token that is required to get the next batch of results.
- If X-2 is >= 0, then A[X-2] is the token that is required to return the last batch of results again.
- A[X-3] is the token that is required to return the batch that comes before the last batch of results again.

Before you use a continuation token to retrieve a previously retrieved batch of results, you must decrease X correctly to ensure that A[X-1] is the continuation token that is required to return the next batch of results.

Example: `GET` method request

This sample code requests a summary of the entries for case comments and detailed audit comments for case folders:

```
#Request to get comment history of a particular case
GET http://example.com:9080/CaseManager/CASEREST/v1/case
/19278CB3-C71C-4DE5-95FE-7C7544020A31/history
?TargetObjectStore=ATOSME&BatchSize=5
&ObjectTypes=CmAcmCaseComment+CmAcmWorkItemComment+
CmAcmVersionSeriesComment&EventTypes=CreationEvent
HTTP/1.1
Host: www.example.net
```

Example: `GET` method response

This sample code shows the summary of the entries for case comments and detailed audit comments for case folders. The fields that are contained in the JSON response differ slightly based on the event types and objects that are included. This example illustrates most of the combinations that you might encounter.

```
#Response
HTTP/1.1 200 OK
Content-Type: application/json;charset-UTF-8
{
  "ContinuationToken":"1,1118.0",
  "Events":
  [
    {
      "EventType":"CreationEvent",
      "EventObjectType":"CmAcmCaseComment",
      "EventTypeLocalizedName":"Creation Event",
      "EventObjectLocalizedName":"Case Comment",
      "EventUser":"P8Admin",
      "EventDateTime":"2010-08-18T18:04:51Z",
      "CmAcmCommentText":"Here is a comment on this case.
        Not my first.",
      "AdditionalProperties":{"CmAcmCaseIdentifier":""}
    },
    {
      "EventType":"CreationEvent",
      "EventObjectType":"CmAcmVersionSeriesComment",
      "EventTypeLocalizedName":"Creation Event",
      "EventObjectLocalizedName":"Version Series Comment",
      "EventUser":"P8Admin",
      "EventDateTime":"2010-08-18T18:00:38Z",
      "CmAcmCommentText":
```

```
      "Test comment for CmAcmVersionSeriesComment.",
  "CmAcmObjectName":"This is the title of the document",
  "AdditionalProperties":{"CmAcmCaseIdentifier":""}
},
{
  "EventType":"CreationEvent",
  "EventObjectType":"CmAcmWorkItemComment",
  "EventTypeLocalizedName":"Creation Event",
  "EventObjectLocalizedName":"Work Item Comment",
  "EventUser":"P8Admin",
  "EventDateTime":"2010-08-18T18:00:16Z",
  "CmAcmCommentText":"This was the right resolution for this
    work item.",
  "CmAcmTaskName":"ETECase1 Task number 1",
  "CmAcmStepName":"test_step_name",
  "AdditionalProperties":{"CmAcmCaseIdentifier":""}
},
{
  "EventType":"CmAcmCaseRelatedEvent",
  "EventTypeLocalizedName":"Case Related Event",
  "EventDateTime":"2011-06-10T22:53:59Z",
  "EventUser":"P8Admin",
  "EventObjectType":"CmAcmCaseFolder",
  "EventObjectLocalizedName":"Case Folder",
  "CmAcmCaseFolder":"{B4CD0C8E-8D1D-424A-A84F-1D2F6F4FB773}",
  "CmAcmRelatedCaseFolder":
      "{56872C9A-D84C-4316-BB49-0EB1062D5F34}",
  "CmAcmCaseTitle":"CTLT_CT1_000000100209",
  "CmAcmRelatedCaseTitle":"CTLT_CT1_000000100212",
  "CmAcmCaseIdentifier":"CTLT_CT1_000000100209",
  "CmAcmRelatedCaseIdentifier":"CTLT_CT1_000000100212",
  "CmAcmRelationshipType":101,
  "Description":
      "split case from poster - test for multi-value",
  "CmAcmCategoryName":null,
  "CmAcmRelatedCaseClassName":"CTLT_CT1",
  "RelatedCaseClassLocalizedName":"CT1",
  "CmAcmObjectName":"000000100209",
  "AdditionalProperties":{"CmAcmCaseIdentifier":"
     CTLT_CT1_000000100209"}
},
{
  "EventType":"CmAcmCaseRelatedEvent",
 "EventTypeLocalizedName":"Case Related Event",
 "EventDateTime":"2012-04-09T14:41:52Z",
 "EventUser":"P8Admin",
 "EventObjectType":"CmAcmCaseFolder",
 "EventObjectLocalizedName":"Case Folder",
 "CmAcmCaseFolder":"{B4CD0C8E-8D1D-424A-A84F-1D2F6F4FB773}",
 "CmAcmRelatedCaseFolder":
     "{56872C9A-D84C-4316-BB49-0EB1062D5F34}",
 "CmAcmCaseTitle":"CTLT_CT1_000000100209",
 "CmAcmRelatedCaseTitle":"CTLT_CT1_000000100212",
 "CmAcmCaseIdentifier":"CTLT_CT1_000000100209",
 "CmAcmRelatedCaseIdentifier":"CTLT_CT1_000000100212",
 "CmAcmRelationshipType":0,
 "Description":
     "relating case 1 to case 2 due to similar victim profile",
 "CmAcmObjectName":"000000100209",
 "CmAcmCategoryName":"victim profile",
 "CmAcmRelatedCaseClassName":"CTLT_CT1",
 "RelatedCaseClassLocalizedName":"CT1",
 "AdditionalProperties":
     {"CmAcmCaseIdentifier":" CTLT_CT1_000000100209"}
},
{
 "EventType":"CmAcmCaseRelatedEvent",
       "EventTypeLocalizedName":"Case Related Event",
       "EventDateTime":"2012-04-09T14:48:29Z",
       "EventUser":"P8Admin",
       "EventObjectType":"CmAcmCaseFolder",
       "EventObjectLocalizedName":"Case Folder",
       "CmAcmCaseFolder":"{B4CD0C8E-8D1D-424A-A84F-1D2F6F4FB773}",
       "CmAcmRelatedCaseFolder":"{56872C9A-D84C-4316-BB49-0EB1062D5F34}",
       "CmAcmCaseTitle":"CTLT_CT1_000000100209",
       "CmAcmRelatedCaseTitle":"CTLT_CT1_000000100212",
       "CmAcmCaseIdentifier":"CTLT_CT1_000000100209",
       "CmAcmRelatedCaseIdentifier":"CTLT_CT1_000000100212",
       "CmAcmRelationshipType":1,
       "Description":"unrelating case 1 from case 2",
       "CmAcmObjectName":"000000100209",
       "CmAcmCategoryName":"victim profile",
       "CmAcmRelatedCaseClassName":"CTLT_CT1",
       "RelatedCaseClassLocalizedName":"CT1",
       "AdditionalProperties":
          {"CmAcmCaseIdentifier":" CTLT_CT1_000000100209"}
      },
      {
       "EventType":"ChangeStateEvent",
       "EventTypeLocalizedName":"Change State Event",
       "EventDateTime":"2012-04-14T01:28:52Z",
       "EventUser":"P8Admin",
       "EventObjectType":"CmAcmCaseTask",
       "EventObjectLocalizedName":"Case Task",
       "CmAcmObjectState":4,
       "CmAcmLastRestartDate":"",
       "CmAcmRestartCount":null,
       "CmAcmDisabledState":0,
       "CmAcmObjectName":"Case1AutomaticTask2"
      }
      ...
  ]
}
```

Related reference:

"Error responses" on page 23

"Case management REST resource URIs" on page 21

"Symbolic names" on page 22

Managing workflows, roles, and in-baskets by using the Process Engine REST Service

You can use the Process Engine REST Service to manipulate the workflow-related aspects of tasks. Specifically, you use this REST Service to access and manage workflows, roles, and in-baskets.

You can use the resources defined in the Process Engine REST Service to perform the following case management tasks:

- Retrieve the contents of an in-basket that are based on the role of a case worker
- Retrieve the workflow step element when the case worker opens a workflow
- View and update the work items in a workflow
- Track workflow processes
- Retrieve the process history for a workflow
- View and update workflow roles, including adding users and groups to roles
- View all assigned work in a case

The Process Engine REST Service provides the following resources that you can use to get case management information.

Table 87. Process Engine REST Service resources

Resource	URI resource name	Description
Application space names	`/appspacenames`	Gets the collection of the names of the application spaces, including the application spaces to which the current user does not have access permissions. You can use this information to select an application space for page creation.
MyRoles	`/appspacenames/{appspace}/myroles`	Gets a collection of roles within an application space.
Role	`/appspacenames/{appspace}/roles/{role}`	Get the role information and in-baskets that are associated with the specified role.
Writeable application space roles	`/writableappspaces/{appSpace}/roles`	Gets the collection of roles defined for an application space to which you can assign members. To access this resource, you must have write access to the application space.
Writeable application space role members	`/writableappspaces/{appspace}/roles/{role}/members`	Gets the set of members that are assigned to a specified role. To access this resource, you must have write access to the application space.
Writeable application space role members update	`/writableappspaces/{appspace}/roles/{role}/members`	Updates the role membership for the specified role. To access this resource, you must have write access to the application space.
Security domains	`/securitydomains`	Gets the names of all the security domains (LDAP realms) found. You can use this information to narrow the scope of users and groups for subsequent operations, such as querying user information for role membership changes.
Users	`/users`	Gets a collection of users from LDAP. You can limit the search scope by using the domainName GET parameter to specify the domain.
Groups	`/groups`	Gets a collection of groups from LDAP. You can use this information to select groups for role memberships or work item assignments. You can limit the search scope by using the domainName GET parameter to specify the domain.
Current® user	`/currentuser`	Gets the name and ID of the user that is currently logged on to the application space.

The following request uses the Process Engine REST Service to return a list of roles that are defined for a solution:

```
http://myserver:9080/CaseManager/P8BPMREST/p8/bpm/v1/appspaces/
Candidate%20Selection%202/myroles?cp=newportvm24_796_tos02
```

The following example shows the format of the response to the request:

```
{
  "Customer Service Representative":
  {
    "name":"Customer Service Representative",
    "URI":"appspaces\/Dannay+Insurance+Claims\/roles\/Customer+Service
      +Representative"
  },
  "Adjuster":
  {
    "name":"Adjuster",
    "URI":"appspaces\/Dannay+Insurance+Claims\/roles\/Adjuster"
  }
}
```

Related reference:

Process Engine REST Service Reference

Managing case folders and documents by using IBM CMIS for FileNet Content Manager

You use the IBM CMIS for FileNet Content Manager in your case management application to create and access the case content that is stored in an object store.

You can use IBM CMIS for FileNet Content Manager to perform tasks such as:
- Querying for cases
- Creating a case
- Updating case properties
- Getting the metadata for case classes and document classes
- Adding a folder in a case folder
- Adding a document to a case

Restriction: You cannot use IBM CMIS for FileNet Content Manager to handle workflow objects and other case-specific objects.

Tip: To access data related to cases and documents, use the JavaScript model APIs that are provided by IBM Content Navigator and IBM Case Manager. These APIs provide more capabilities than are available in IBM CMIS for FileNet Content Manager.

Using IBM CMIS for FileNet Content Manager to create a case

There are two ways in which you can use IBM CMIS for FileNet Content Manager to create a case and its folder hierarchy. You can create a case:
- Automatically by creating a document of one of a specific set of document classes anywhere in the target object store
- Manually by using IBM CMIS for FileNet Content Manager to create the case folder directly under the case type folder

Creating a case by creating a document

You might want to create a case automatically whenever a certain type of document is received. For example, whenever a new loan application is received, you might want to create the case that is used to process that application.

By using IBM CMIS for FileNet Content Manager, you can specify that a case is to be created whenever an instance of a specified document class is created. When the document is created, the Content Platform Engine event handler automatically names the case folder according to IBM Case Manager naming requirements and places it in the appropriate folder under the case type.

In Case Manager Builder, you can specify that whenever a document of a specific document class is created anywhere in the target object store, a case folder is created automatically under a specific case type of the target solution. The document is then filed as a child of the new case folder and the `CmAcmInitiatingDocument` property of the new case folder is set to reference the document.

It does not matter where in the folder hierarchy the document is created. When IBM CMIS for FileNet Content Manager creates the case folder, it automatically files the document referentially as a child folder.

Creating a case by creating the case folder

Tip: The preferred method for creating a case folder is to use the cases resource provided by the IBM Case Manager REST protocol.

You can use IBM CMIS for FileNet Content Manager to enable a case worker to create a case by creating the case folder as an immediate child of the case type folder. The properties that you specify for the case folder become the properties of the case. The case properties are global to all the tasks in the case and to all the workflows of those tasks.

The system automatically renames the case folder by assigning the next available sequence number for the case type as the new folder name. The system then moves the new folder to the bottom of the `*case type folder*/Cases/yyyy/mm/dd/hh` subfolder hierarchy, creating any missing parts of the `yyyy/mm/dd/hh` subhierarchy as necessary.

Related reference:

"Getting and changing case information" on page 52

"Cases resource" on page 53

Related information:

IBM CMIS for FileNet Content Manager Development

Configuring a solution to create a case when a document is added to the object store

You can configure your solution to create cases programmatically when a document is added to the target object store. You use the initiating document setting when you create the case type in Case Manager Builder, and then configure the addition of the document to the object store in one of several ways.

When you add the solution in Case Manager Builder, you specify the properties of each case type. For a solution that will include starting cases when documents are added to the object store, you must first specify that setting when you configure the case type.

After you create the solution in Case Manager Builder, you configure the applications or workflows to add the document classes that will initiate cases in the deployed solution.

Content Engine Java API
: You can use the Content Engine Java API to develop a custom application that receives the document from the user. The application uses the API to set the document class to the correct starting document class, and to check the document into the object store. The checkin of the document triggers the IBM Case Manager event handler to create a case.

CE_Operations
: You can define a workflow that is either internal or external to the solution. Include the CE_Operations create document step in the workflow. The step creates a document that has the same document class as the starting document for the case type. When the CE_Operation step is executed, a case is automatically created by the IBM Case Manager event handler.

External applications, such as Datacap Studio
: Configure an external application, such as Datacap Studio, to inject documents into the target object store. The documents must have the type the same as the starting document class for the case. When the document is created in the object store, the case is automatically created by the IBM Case Manager event handler.

Getting case data from an external data source

IBM Case Manager stores case data in Content Platform Engine. However, you can use an external data service with a solution to access data from a different repository or other data source. This data is then incorporated into the case and stored with the rest of the case data in Content Platform Engine.

About this task

For example, you might have a database that contains detailed customer records. When a case worker enters a customer's serial number, the external data service can get the name and address of the customer from that database. These values are then incorporated into the case data and stored in Content Platform Engine.

In addition to getting property values, you can use an external data source to modify property attributes such as minimum value or maximum value. The external data service must work within any constraints placed on the property attributes in Content Platform Engine. For example, if a minimum value is specified for the property in Content Platform Engine, the external data 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.

You can also use the external data service to define dependencies between properties. By using this feature, you can implement dynamic behavior in your solution. For example, you might specify a dependency between a **`state`** property and a **`city`** property. When a case worker selects a state, the choice list that is associated with the **`city`** property contains only cities that are in that state.

Procedure

To get data from an external data source:

1. Use the IBM Case Manager APIs to implement a service to extract case data from the external data source.
2. Use the IBM Case Manager configuration tool to register the external data service for use with your solution.

   Restriction: You can register only one external data service for a solution.
3. Deploy or redeploy the solution.

Results

After you register the external data service, Case Manager Client communicates with the service to get case data whenever case workers create cases or modify cases. This communication is handled automatically through the IBM Case Manager APIs.

For properties that are associated with an external data service choice list, only the value, not the display name, is persisted IBM Case Manager. The Search widget, which generates a result set for the Case List widget, does not call the external data service to retrieve the display names for these properties. However, selecting the case from the Case List widget does cause the external data service to retrieve the property display names.

Important: You use the external data service only for retrieving data from an external source. For example, when a case worker creates or modifies a case, Case Manager Client saves the data that was received from the external data source in Content Platform Engine. If the case worker modifies this data, Case Manager Client does not update the corresponding data in the external data source.

"Implementing an external data service by using the REST protocol"

Implementing an external data service by using the REST protocol

To use external data with your solution, you must create a service that implements the external data service REST protocol that is provided with IBM Case Manager. This protocol provides for the communication between Case Manager Client and the external data source.

About this task

In addition to implementing the REST protocol, the external data service must implement any authentication that is required by the external data source.

Procedure

To implement an external data service:

1. Implement the `POST` method for the particular object type resource. This method is called automatically by the IBM Case Manager REST protocol in response to requests from Case Manager Client to create or modify a case. The external data service must submit the data for the case properties that it manages back to the IBM Case Manager REST protocol in the response to the `POST` method.
2. If the service modified any attributes for the case properties that it manages, retrieve the property attributes. To retrieve property attributes, you can use the Content Engine Java protocol or IBM CMIS for FileNet Content Manager in the external data service.
3. If the external data service needs to authenticate users, configure authentication for users.

"Particular object type resource"

"Authentication for external data services" on page 113

"Persistence of case data" on page 114

"Example data flow for case creation" on page 114

Particular object type resource

The particular object type resource represents a case type in which property values are obtained from an external data source. When a case of the specified case type is created or modified, the IBM Case Manager REST protocol uses this resource to obtain data for that case from the external data source.

You do not use this resource directly in your case management application. Instead, the IBM Case Manager REST protocol calls the `POST` method for resource automatically when a case is being added or modified to communicate with the external data service. The service then returns the required information in the response to this method call.

"POST method for the particular object type resource" on page 103

"Request modes" on page 106

"Client context for work items" on page 107

"Response to a request for case data" on page 108

"Error responses for an external data service" on page 113

POST method for the particular object type resource

The POST method provides the means for obtaining data from an external data source for a case of a specific case type. You do not call this method directly. Instead, the IBM Case Manager REST protocol calls this method automatically when a case is being added or modified.

When the IBM Case Manager REST protocol calls the POST method, the request payload contains the current value for each case 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 case worker 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.

The IBM Case Manager REST protocol then merges these changes into the case data and returns the data to the Case Manager Client.

URI syntax

/type/{*object type name*}

The URI for the POST method includes the following path element:

Table 88. Path elements for the POST method

Name	Type	Description
{object type name}	String	The symbolic name of case type that defines the case that is being updated.

Request content

```
{
   "repositoryId":"<target object store name>",
   "objectId"   : "<GUID of the case folder>",
   "requestMode" : "<request context>",
   "externalDataIdentifier" : "<identifier for service">,

   "properties":
   [
   {
      "symbolicName" : "<property name>",
      "value"        : <current value>,
   }

   // More properties ...

   ],

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

      }
```

Table 89. Request parameters for the POST method

Name	Type	Required?	Description
repositoryId	String	Yes	The symbolic name of the object store that contains the case type. A symbolic name is called a unique identifier in IBM Case Manager.
objectId	String	No	The GUID that identifies the root folder of an existing case. This parameter is not specified when the POST method is called to create a case.
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
externalData Identifier	String	Yes, for certain **request Mode** settings	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 case type. For each property, the request contains the symbolic name and the property value.
clientContext	Array	No	An array that contains a series of key value pairs that specify contextual information for a specific work item. This parameter is used to send information to an external data service when a case worker opens the work item.

Response codes

Table 90. Response codes for the POST method

Code	Description
200 OK	The method completed successfully. The response that is returned by the POST method includes the updated information for the case.
400 Bad Request	One of the required parameters was missing or a parameter value was invalid.
404 Not Found	The case type that was specified in the request was not found. This error does not indicate that the case type is invalid. Instead, it indicates that the external data service does not manage any property values for the case type. The IBM Case Manager REST protocol does not return an error to the Case Manager Client.
500 Internal Server Error	A server error occurred. For information about the error, see the userMessage element in the JSON response.

Example: POST method request

This sample code submits a request to an external data service when a case worker selects a value for the state property, **DH2_State**. The service then updates the choice list for the city property, **DH2_City**, which depends on the **State** property.

```
POST /testservice/ICMEDREST/type/DH2_MyCase
{
  "repositoryId": "CMTOSDH",
  "requestMode": "inProgressChanges",
  "externalDataIdentifier": "-1,0",
  "properties": [

    // Non-external data related properties

    {
      "symbolicName": "CmAcmCaseIdentifier",
      "value": null
    },
    {
      "symbolicName": "CmAcmCaseState",
      "value": 0
    },

    // ...

    {
      "symbolicName": "DH2_State",
      "value": "CA"
    },
    {
      "symbolicName": "DH2_PropOne",
      "value": null
    },
    {
      "symbolicName": "DH2_MVInt",
      "value": [
        0,
        100
      ]
    },
    {
      "symbolicName": "DH2_MVString",
      "value": [ ]
    },
    {
      "symbolicName": "DH2_City",
      "value": null
    }
  ]
}
```

Example: `POST` method response

This sample code shows the information that is returned by the external data service when a case worker selects a value for the state property, **`DH2_State`**:

```
{
  "externalDataIdentifier": "1,0",
  "properties": [
    {
      "symbolicName": "DH2_City",
      "hidden": false,
      "required": true,
      "hasDependentProperties": false,
      "choiceList": {
        "displayName": "CityChoiceList",
        "choices": [
          {
            "displayName": "Los Angeles",
            "value": "Los Angeles"
          },
          {
```

```
          "displayName": "San Diego",
          "value": "San Diego"
        },
        {
          "displayName": "San Francisco",
          "value": "San Francisco"
        }
      ]
    }
  }
 ]
}
```

Related reference:

"Request modes"

"Common JSON payload for cases and case types" on page 23

"Response to a request for case data" on page 108

"Client context for work items" on page 107

Request modes

When a case is created or modified in Case Manager Client, the IBM Case Manager REST protocol calls the `POST` method for the particular object type resource to submit a request to the external data service. This request 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 create a case, the service needs to respond with the initial property values that are defined for the case type.

The **`requestMode`** parameter indicates the action that is being performed in Case Manager Client. This action determines the response that is returned by the external data service.

The **`requestMode`** parameter can have the following values:

initialNewObject
: This value indicates that the external data service is being called for the first time in a sequence of exchanges to create a case. For each property, the input payload contains the symbolic name and the default value that is defined in the case type.

  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 creation of the case include the **`externalDataIdentifier`** parameter to indicate the current state of the data to the service.

initialExistingObject
: Indicates that the external data service is being called for the first time in a sequence of exchanges to modify an existing case. 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 GUID of the root folder for the case. The service can use this GUID to refer to the case. However, remember that the values stored in the repository for the case 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 case.

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 case 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. The input payload can contain the following information:

- The current working value for each property in the case.
- The **externalDataIdentifier** parameter, which indicates to the service the previous state of any properties that it updated.
- For an existing case, the **objectId** parameter, which specifies the GUID of the root folder for the case.

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
Indicates that the external data service is being called for the final time in the sequence of exchanges to create a case. After this call, the new case 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 case type.

finalExistingObject
Indicates that the external data service is being called for the final time in the sequence of exchanges to update an existing case. After this call, the updated case 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 case type.

Client context for work items

The **clientContext** parameter provides contextual information about a work item that a case worker opened. An external data service can use this information to determine the appropriate response. For example, an account identifier might typically be read-only. However, if the work item is to open an account, the external data service can set the account identifier to be writable.

For the IBM Case Manager widgets, Case Manager Client automatically includes the **clientContext** parameter in the request when a case worker opens a work item.

The client context is defined by the **clientContext** parameter. The following table describes the keys that this parameter can contain.

Tip: Unless otherwise stated in the table, you can obtain a value for a key by using the Process Engine REST service to query the `systemProperties` object for the step.

Table 91. Keys in the **`clientContext`** *parameter*

Key	Data type of value	Description
connectionPoint	String	The name of the isolated region in the workflow system database that contains the workflow definition for the task. For a custom widget, you can obtain this value by querying the solution space attributes.
stepId	Integer	The identifier of the step.
mapName	String	The name of map that the work item locates.
workflowNumber	String	The unique identifier that is assigned to the workflow that is associated with the task.
workflowName	String	The name of the workflow that is associated with the task.
caseTaskId	String	The unique identifier that is assigned to the task in this case.
stepName	String	The name of current step.
workObjectNumber	String	The unique identifier that is assigned to the work item.
authoredMapName	String	The map name according to the current locale of the user.
queueName	String	The name of the queue with which the work item is associated. For a custom widget, you can obtain this value by querying the in-basket attributes.
originator	String	The identifier assigned to the case worker who launch the work flow.
subject	String	The workflow subject of current task. For an automatic task or a manual task, the subject is the name assigned to the task in Case Manager Builder. For a user-created task, the subject is the name that the case worker assigned to the task.
launchDate	DateTime	The date that the workflow was launched.
role	String	The role that the case worker is currently using. For a custom widget, you can obtain this value by querying the in-basket attributes.

Related reference:

"Cases resource" on page 53

"Particular case type resource" on page 44

"Particular case instance resource" on page 56

"Common JSON payload for cases and case types" on page 23

"POST method for the particular object type resource" on page 103

Response to a request for case data

The external data service responds to a `POST` method that was submitted by the IBM Case Manager REST API. 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:

```
{
  "externalDataIdentifier" : "<opaque identifier meaningful to service>",

  "properties":
  [
    {
      "symbolicName"   : "<symbolic_name>",
      "value"           : <potential new value>,
      "customValidationError" : "Description of an invalid reason",
      "customInvalidItems" : [0,3,4,8], // invalid multi-value items
      "displayMode"     : "<readonly/readwrite>",
      "required"      : <true or false>,
      "hidden"          : <true or false>,
      "maxValue"      : <overridden max value>,
      "minValue"      : <overridden min value>,
      "maxLength"      : <underlying max>,

      "choiceList"   :
      {
        "displayName"   : "<display_name>",
        "choices"      :
        [
          {
            "displayName"   : "<name>",
            "value"   : <value>
          },

          {
            "displayName"     : "<name>",
            "value"   : <value>
          },

          // More choices ...
        ]
      }

      "hasDependentProperties" : <true or false>,

    }

    // More properties ...

  ]

}
```

Table 92. Response parameters for the `POST` method

Name	Type	Required?	Description
`externalData Identifier`	String	Yes	The identifier provides contextual information to indicate the state of the data that the service is returning. You implement this parameter with values that are meaningful for your data source. Typically, the parameter references the specific configurations that define attributes other than the property value such as the minimum value, maximum value, or the choice list. These configurations can be selected dynamically based on other property values. In this situation, the service can use the **`externalDataIdentifier`** parameter to determine that the configuration changed since the previous call. If the external data service does not modify property attributes dynamically, you might not need to capture the data state. In this situation, you might implement the parameter to return a fixed string value. If the external data service returns data that is dynamic, you must capture the data state. For example, the service might manage a property whose value or other attributes are determined by the value of another property. In this situation, you must implement the parameter to return a value that references the specific configuration that was used to determine the value or other attributes of the dependent property. The parameter must capture enough information to identify changes in property data when the **`externalDataIdentifier`** parameter is returned to the service in an **`inProgressChanges`** request. For example, assume that a list of conditions is used to select a set of configurations based on the working property values. The data that is captured in the **`externalDataIdentifier`** parameter might include the indexes of the matching conditions.
`properties`	Array	Yes	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.

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 93. Attributes of properties in the response payload

Name	Type	Required?	Description
`symbolicName`	String	Yes	The symbolic name of the property. The name must match the symbolic name that was specified in the request payload.
`value`	Determined by setting in the case type	No	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 case type. The external data service can determine the property value dynamically based on the values of another other property. If the service does not specify a value, the current working value for the property is unchanged.

Table 93. Attributes of properties in the response payload (continued)

Name	Type	Required?	Description
custom Validation Error	String	No	A message that describes why a property value is invalid. 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 value unchanged and return an error message in the **customValidationError** parameter. For example, the service might determine that an account number is invalid. However, you do not want the service to replace the account number. Instead, you can configure the service to return an error message in the **customValidationError** parameter. If this parameter is included in the response, the property value is deemed invalid. However, the absence of this attribute indicates only that the parameter passed the validation by the external data service. The value might still be invalid based on attributes that are not validated by the service.
custom Invalid Items	Array of indexes	No	An array of indexes for a list of values for a multi-valued property. When the external data service validates a multi-valued property, it can return this parameter to indicate the specific values that are invalid. If a multi-valued property is invalid and this parameter is not set, the property value as a whole is considered invalid. This attribute is applicable only if the **customValidationError** parameter indicates that the property is invalid.
displayMode	String	No	A string that specifies whether Case Manager Client is to display the property value as read-only. The external data service can set this parameter to one of the following values: **readonly** The user can view the property value but cannot modify it. **readwrite** The case worker can modify the property value. This setting is the default value. If the property value is set to `readonly` in the case type, the external data service cannot make the value writable. In this situation, a value of `readwrite` is ignored.
required	Boolean	No	A Boolean value that is set to true to indicate that a value is required for the property. The external data service can determine this setting dynamically based on the values of other properties. However, the service cannot override the **required** parameter if it is set to true in the case type.
hidden	Boolean	No	A Boolean value that is set to true to indicate that the property is to be hidden in Case Manager Client. The external data service can determine this setting dynamically based on the values of other properties. If this parameter is not specified, the value specified in the case type is used.

Table 93. Attributes of properties in the response payload (continued)

Name	Type	Required?	Description
maxValue	Integer, float, or date-time	No	A number that indicates the maximum value of the property. The external data service can determine this setting dynamically based on the values of other properties. If a maximum value is specified for the property in Content Platform Engine, 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 Content Platform Engine is 100, the service can set the value to 50, but not to 150.
minValue	Integer, float, or date-time	No	A number that indicates the minimum value of the property. The external data service can determine this setting dynamically based on the values of other properties. If a minimum value is specified for the property in Content Platform Engine, 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 Content Platform Engine is 100, the service can set the value to 150, but not to 50.
maxLength	Integer	No	A number that indicates the maximum length of the property value. The external data service can determine this setting dynamically based on the values of other properties. If a maximum length is specified for the property in Content Platform 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 Platform Engine is 100, the service can set the value to 50, but not to 150.

choiceList	Object	No	An array that defines a list of choices for the property value.
			
			The external data service can specify a choice list only if one is not defined for the property in Content Platform Engine. The service can determine the choices in the list dynamically based on the values of other properties.
			
			The **choiceList** value can contain a flat list of choices:
			`"choiceList" :`
			`{`
			`  "displayName" : "<display name for the choice list>",`
			`  "choices" :`
			`  [`
			`    {`
			`      "displayName" : "<display name for a specific choice>`
			`      "value" : <value>`
			`    },`
			
			`    {`
			`      "displayName" : "<display name for a specific choice>",`
			`      "value" : <value>`
			`    },`
			
			`    // More choices ...`
			`  ]`
			`}`

Table 93. Attributes of properties in the response payload (continued)

Name	Type	Required?	Description
hasDependentProperties	Boolean	No	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. By default, this parameter is set to false.

Error responses for an external data service

If the POST method call fails, the response code that the IBM Case Manager REST protocol returns indicates the type of error that occurred.

For example, the response code 404 Not Found indicates that the method did not find a resource, such as the specified solution or case 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",
    ]
  }
}
```

Authentication for external data services

If your external data service needs to authenticate users, it must participate in the same single sign-on authentication configuration as the other IBM Case Manager components, such as Case Manager Client or the IBM Case Manager REST protocol.

If Content Platform Engine and the external data service do not use the same WebSphere Application Server profile, you must set up Lightweight Third Party Authentication (LTPA) security between the applications in WebSphere Application Server. Begin by exporting the LTPA key from the Content Platform Engine server.

The IBM Case Manager REST protocol passes one of the following headers to the external data service:

Basic If basic authentication is used, the protocol passes an authorization header that contains the keyword **Basic** that is followed by the encoded user name and password pair.

LtpaToken2

If LTPA authentication is used, the protocol passes an LTPA token with the cookie **LtpaToken2**.

If the request contains either of these authentication values, WebSphere Application Server first authenticates with the LDAP server, if one is configured. WebSphere Application Server then sets up a JAAS subject in the calling context of the external data service. To retrieve this JAAS subject, you can use one of the WebSphere Application Server Java APIs. Alternatively, you can use the helper method `javax.security.auth.Subject getAmbientSubject( )` that is defined for the `UserContext` class in the Content Engine Java API.

Persistence of case data

When a case worker saves a new case or an updated case, the IBM Case Manager REST protocol makes a final call to the external data service. The REST protocol uses the response returned by the service to determine which values are to be saved to the repository for the case.

For the final call to the external data service, the **requestMode** parameter is set to `finalNewObject` for a new case or `finalExistingObject` for an updated case.

The IBM Case Manager REST protocol evaluates each property as follows:

- A value that is explicitly passed to the protocol by Case Manager Client is checked against any property attributes that are returned by the external data service. If the value passes validation, that value is saved for the case in Content Platform Engine. If the value does not pass validation, the protocol returns an error.
- If a value is not explicitly passed by Case Manager Client and the service passes a value for the property, that value is saved for the case in Content Platform Engine.
- If a value is not explicitly passed by Case Manager Client and the service does not pass a value, the default value for a new case is saved for the case in Content Platform Engine. For an existing case, the current value is unchanged. However, the REST protocol checks the default value or current value against any property values returned by the service. If the value does not pass validation, the REST protocol returns an error.

Typically, if the **requestMode** parameter is set to `finalNewObjector` `finalExistingObject`, the external data service overrides the current working value of a property if that value is invalid. Typically, the service validates property values in earlier requests, so this situation rarely occurs. However, you can implement the service to override invalid property values. In particular, the service can override a property value when the display mode is set to read-only. The case worker cannot change the property value in this situation.

Example data flow for case creation

If an external data service is registered for a solution, data for a new case is automatically obtained from that service. The IBM Case Manager REST protocol handles the exchange of data between Case Manager Client and the external data service.

"Retrieval of initial information for a new case" on page 115

"Update of a property that has dependencies" on page 120

"Creation of the new case" on page 123

Retrieval of initial information for a new case

The first step in creating a case is to retrieve the properties that are defined for the case type. As part of this process, the IBM Case Manager REST API obtains data from the external data service for any properties that the service manages.

The following steps show the flow of data in the retrieval of the properties for a case type called *DH2_MyCase*:

1. The case worker clicks **Add Case** and selects the appropriate case type.
2. Case Manager Client submits a request to the IBM Case Manager REST API to obtain a complete list of the case properties and their attributes, including their default values. The request is submitted by calling the GET method of the particular case type resource:
   ```
   GET /CaseManager/CASEREST/v1/casetype/DH2_MyCase
   ```
3. The IBM Case Manager REST API passes the default case data to the external data service by calling the POST method particular object type resource:
   ```
   POST /testservice/ICMEDREST/type/DH2_MyCase
   {
     "repositoryId": "CMTOSDH",
     "requestMode": "initialNewObject",
     "properties": [

       // Payload may include additional properties
       // not meaningful to the external data service

       {
         "symbolicName": "CmAcmCaseIdentifier",
         "value": null
       },
       {
         "symbolicName": "CmAcmCaseState",
         "value": 0
       },

       // ...

       {
         "symbolicName": "DH2_State",
         "value": null
       },
       {
         "symbolicName": "DH2_PropOne",
         "value": null
       },
       {
         "symbolicName": "DH2_MVInt",
         "value": [ ]
       },
       {
         "symbolicName": "DH2_MVString",
         "value": [ ]
       },
       {
         "symbolicName": "DH2_City",
         "value": null
       }
     ]
   }
   ```
4. The external data service responds with the changes to the attributes for the properties that it manages. The response also includes the initial setting for the external data identifier.

```
{
  "externalDataIdentifier": "-1,0",
  "properties": [
    {
      "symbolicName": "DH2_State",
      "required": true,
      "maxLength": 2,
      "hasDependentProperties": true,
      "choiceList": {
        "displayName": "StateChoiceList",
        "choices": [
          {
            "displayName": "New York",
            "value": "NY"
          },
          {
            "displayName": "California",
            "value": "CA"
          },
          {
            "displayName": "Nevada",
            "value": "NV"
          }
        ]
      }
    },
    {
      "symbolicName": "DH2_PropOne",
      "maxValue": 10,
      "minValue": 1,
      "hasDependentProperties": false
    },
    {
      "symbolicName": "DH2_MVInt",
      "value": [
        0,
        100
      ],
      "maxValue": 1000,
      "minValue": 0,
      "hasDependentProperties": true
    },
    {
      "symbolicName": "DH2_MVString",
      "required": true,
      "maxLength": 24,
      "hasDependentProperties": false,
      "choiceList": {
        "displayName": "MVStringChoiceList",
        "choices": [
          {
            "displayName": "One",
            "value": "One"
          },
          {
            "displayName": "Two",
            "value": "Two"
          },
          {
            "displayName": "Three",
            "value": "Three"
          },
          {
            "displayName": "Ten",
            "value": "Ten"
          },
          {
            "displayName": "Eleven",
            "value": "Eleven"
          },
          {
            "displayName": "Twelve",
            "value": "Twelve"
          }
        ]
      }
    },
    {
      "symbolicName": "DH2_City",
      "value": null,
      "displayMode": "readonly",
      "hidden": true,
      "required": true,
      "hasDependentProperties": false
    }
  ]
}
```

5. The IBM Case Manager REST API merges this information with the default case data and returns the updated values to Case Manager Client:

```
{
  "Properties": [

    // Non-external data related properties

    {
      "SymbolicName": "CmAcmCaseIdentifier",
      "DisplayName": "Case Identifier",
      "Value": null,
      "DisplayMode": "readwrite",
      "Description": "A specially formatted identifier for Case Folder
         instances, consists of Case Folder subclass symbolic class name,
         \"_\" and then a 12 digit sequence number with leading zeros.",
      "PropertyType": "string",
      "Cardinality": "single",
      "Updatability": "readwrite",
      "Required": false,
      "Queryable": true,
      "Orderable": true,
      "Hidden": false,
      "Inherited": true,
      "DefaultValue": null,
      "MaxLength": 85,
      "HasDependentProperties": false
    },
    {
      "SymbolicName": "CmAcmCaseState",
      "DisplayName": "Case State",
      "Value": 0,
      "DisplayMode": "readwrite",
      "Description": "An integer choice property that defines
         the possible states of Case Folder instance.",
      "PropertyType": "integer",
      "Cardinality": "single",
      "Updatability": "readwrite",
      "Required": true,
      "Queryable": true,
      "Orderable": true,
      "Hidden": false,
      "Inherited": true,
      "DefaultValue": 0,
      "MaxValue": null,
      "MinValue": null,
      "ChoiceList": {
```

```
      "DisplayName": "CmAcmCaseStateChoiceList",
      "Choices": [
        {
          "ChoiceName": "New",
          "Value": 0
        },
        {
          "ChoiceName": "Initializing",
          "Value": 1
        },
        {
          "ChoiceName": "Working",
          "Value": 2
        },
        {
          "ChoiceName": "Complete",
          "Value": 3
        },
        {
          "ChoiceName": "Failed",
          "Value": 4
        }
      ]
    },
    "HasDependentProperties": false
  },

  // ...

  {
    "SymbolicName": "DH2_State",
    "DisplayName": "State",
    "Value": null,
    "DisplayMode": "readwrite",
    "Description": "State where home office is located",
    "PropertyType": "string",
    "Cardinality": "single",
    "Updatability": "readwrite",
    "Required": true,
    "Queryable": true,
    "Orderable": true,
    "Hidden": false,
    "Inherited": false,
    "DefaultValue": null,
    "MaxLength": 2,
    "ChoiceList": {
      "DisplayName": "StateChoiceList",
      "Choices": [
        {
          "ChoiceName": "New York",
          "Value": "NY"
        },
        {
          "ChoiceName": "California",
          "Value": "CA"
        },
        {
          "ChoiceName": "Nevada",
          "Value": "NV"
        }
      ]
    },
    "HasDependentProperties": true
  },
  {
    "SymbolicName": "DH2_PropOne",
    "DisplayName": "Prop One",
    "Value": null,
    "DisplayMode": "readwrite",
    "Description": "An integer property",
    "PropertyType": "integer",
    "Cardinality": "single",
    "Updatability": "readwrite",
    "Required": false,
    "Queryable": true,
    "Orderable": true,
    "Hidden": false,
    "Inherited": false,
    "DefaultValue": null,
    "MaxValue": 10,
    "MinValue": 1,
    "HasDependentProperties": false
  },
  {
    "SymbolicName": "DH2_MVInt",
    "DisplayName": "MVInt",
    "Value": [
      0,
      100
    ],
    "DisplayMode": "readwrite",
    "Description": "Multi-value integer property",
    "PropertyType": "integer",
    "Cardinality": "multi",
    "RequiresUniqueElements": false,
    "Updatability": "readwrite",
    "Required": false,
    "Queryable": true,
    "Orderable": false,
    "Hidden": false,
    "Inherited": false,
    "DefaultValue": null,
    "MaxValue": 1000,
    "MinValue": 0,
    "HasDependentProperties": true
  },
  {
    "SymbolicName": "DH2_MVString",
    "DisplayName": "MVString",
    "Value": [ ],
    "DisplayMode": "readwrite",
    "Description": "Multi-value string property",
    "PropertyType": "string",
    "Cardinality": "multi",
    "RequiresUniqueElements": false,
    "Updatability": "readwrite",
    "Required": true,
    "Queryable": true,
    "Orderable": false,
    "Hidden": false,
    "Inherited": false,
    "DefaultValue": null,
    "MaxLength": 24,
    "ChoiceList": {
      "DisplayName": "MVStringChoiceList",
      "Choices": [
        {
          "ChoiceName": "One",
          "Value": "One"
        },
        {
          "ChoiceName": "Two",
          "Value": "Two"
        },
          {
            "ChoiceName": "Three",
            "Value": "Three"
          },
          {
            "ChoiceName": "Ten",
            "Value": "Ten"
          },
          {
            "ChoiceName": "Eleven",
            "Value": "Eleven"
          },
          {
            "ChoiceName": "Twelve",
            "Value": "Twelve"
          }
        ]
      },
      "HasDependentProperties": false
    },
    {
      "SymbolicName": "DH2_City",
      "DisplayName": "City",
      "Value": null,
      "DisplayMode": "readonly",
      "Description": "City where home office is located",
      "PropertyType": "string",
      "Cardinality": "single",
      "Updatability": "readwrite",
      "Required": true,
      "Queryable": true,
      "Orderable": true,
      "Hidden": true,
      "Inherited": false,
      "DefaultValue": null,
      "MaxLength": 64,
      "HasDependentProperties": false
    }
  ],
  "CaseType": "DH2_MyCase",
  "Description": "A simple case type",
  "CaseTitleProperty": "CmAcmCaseIdentifier",
  "DisplayName": "My Case",
  "ExternalDataIdentifier": "-1,0"
}
```

Update of a property that has dependencies

The new case is displayed with the property values that are returned by the IBM Case Manager REST protocol. The case worker then edits the values as needed. If the case worker changes the value of a property that has dependent properties, another call is made to the external data service to update the values of the dependent properties.

The following steps show the flow of data when a case worker updates properties for a new case of type DH2_MyCase:

1. The user selects a value for the **`State`** property on which the **`City`** property depends.
2. Case Manager Client calls the `POST` method of the particular case type resource to submit the value to the IBM Case Manager REST protocol. The request payload includes the working property values, including the value that is selected for the **`State`** property:

```
POST /CaseManager/CASEREST/v1/casetype/DH2_MyCase
{
  "TargetObjectStore": "CMTOSDH",
  "ExternalDataIdentifier": "-1,0",
  "Properties": [

    // Properties not related to external data

    {
      "SymbolicName": "CmAcmCaseIdentifier",
      "Value": null
    },
    {
      "SymbolicName": "CmAcmCaseState",
      "Value": 0
    },

    // ...

    {
      "SymbolicName": "DH2_State",
      "Value": "CA"
    },
    {
      "SymbolicName": "DH2_PropOne",
      "Value": null
    },
    {
      "SymbolicName": "DH2_City",
      "Value": null
    },
    {
      "SymbolicName": "DH2_MVInt",
      "Value": [
        0,
        100
      ]
    },
    {
      "SymbolicName": "DH2_MVString",
      "Value": [ ]
    }
  ]
}
```

3. The IBM Case Manager REST protocol passes the values to the external data service by calling the POST method for the particular object type resource:

```
POST /testservice/ICMEDREST/type/DH2_MyCase
{
  "repositoryId": "CMTOSDH",
  "requestMode": "inProgressChanges",
  "externalDataIdentifier": "-1,0",
  "properties": [

    // Non-external data related properties

    {
      "symbolicName": "CmAcmCaseIdentifier",
      "value": null
    },
    {
      "symbolicName": "CmAcmCaseState",
      "value": 0
    },

    // ...
    {
      "symbolicName": "DH2_State",
      "value": "CA"
    },
    {
      "symbolicName": "DH2_PropOne",
      "value": null
    },
    {
      "symbolicName": "DH2_MVInt",
      "value": [
        0,
        100
      ]
    },
    {
      "symbolicName": "DH2_MVString",
      "value": [ ]
    },
    {
      "symbolicName": "DH2_City",
      "value": null
    }
  ]
}
```

4. The external data service responds with the choice list options for the **City** property based on the selected state. The response also includes an updated value for the external data identifier:

```
{
  "externalDataIdentifier": "1,0",
  "properties": [
    {
      "symbolicName": "DH2_City",
      "hidden": false,
      "required": true,
      "hasDependentProperties": false,
      "choiceList": {
        "displayName": "CityChoiceList",
        "choices": [
          {
            "displayName": "Los Angeles",
            "value": "Los Angeles"
          },
          {
            "displayName": "San Diego",
            "value": "San Diego"
          },
          {
            "displayName": "San Francisco",
            "value": "San Francisco"
          }
        ]
      }
    }
  ]
}
```

5. The IBM Case Manager REST protocol merges the changes from the external data service into the working values for the case and returns the updated values to the Case Manager Client:

```
{
  "Properties": [
    {
      "SymbolicName": "DH2_City",
      "DisplayName": "City",
      "Value": null,
        "DisplayMode": "readwrite",
        "Description": "City where home office is located",
        "PropertyType": "string",
        "Cardinality": "single",
        "Updatability": "readwrite",
        "Required": true,
        "Queryable": true,
        "Orderable": true,
        "Hidden": false,
        "Inherited": false,
        "DefaultValue": null,
        "MaxLength": 64,
        "ChoiceList": {
          "DisplayName": "CityChoiceList",
          "Choices": [
            {
              "ChoiceName": "Los Angeles",
              "Value": "Los Angeles"
            },
            {
              "ChoiceName": "San Diego",
              "Value": "San Diego"
            },
            {
              "ChoiceName": "San Francisco",
              "Value": "San Francisco"
            }
          ]
        },
        "HasDependentProperties": false
      }
    ],
    "CaseType": "DH2_MyCase",
    "ExternalDataIdentifier": "1,0"
  }
```

Creation of the new case

After the case worker enters the data for the case, the final step in creating a case is to add the case to the repository. The external data service is called again to validate the data.

The following steps show the flow of data when a case worker adds a case to the repository:

1. The case worker finishes updating the case properties and clicks **Add**.
2. Case Manager Client submits the working property values to the IBM Case Manager REST protocol by calling the `POST` method of the cases resource:

```
POST /CaseManager/CASEREST/v1/cases
{
  "TargetObjectStore": "CMTOSDH",
  "CaseType": "DH2_MyCase",
  "ExternalDataIdentifier": "1,1",
  "Properties": [

    // Non-external data related properties

    {
      "SymbolicName": "CmAcmCaseIdentifier",
      "Value": null
    },
    {
      "SymbolicName": "CmAcmCaseState",
      "Value": 0
    },
    // ...

    {
      "SymbolicName": "DH2_State",
      "Value": "CA"
    },
    {
      "SymbolicName": "DH2_PropOne",
      "Value": 7
    },
    {
      "SymbolicName": "DH2_City",
      "Value": "San Diego"
    },
    {
      "SymbolicName": "DH2_MVInt",
      "Value": [
        0,
        101,
        200,
        210
      ]
    },
    {
      "SymbolicName": "DH2_MVString",
      "Value": [
        "One",
        "Three",
        "Thirty"
      ]
    }
  ]
}
```

3. The IBM Case Manager REST protocol submits the property values to the external data service by calling the POST method of the particular object type resource:

```
POST /testservice/ICMEDREST/type/DH2_MyCase
{
  "repositoryId": "CMTOSDH",
  "requestMode": "finalNewObject",
  "externalDataIdentifier": "1,1",
  "properties": [

    // Non-external data related properties

    {
      "symbolicName": "CmAcmCaseIdentifier",
      "value": null
    },
    {
      "symbolicName": "CmAcmCaseState",
      "value": 0
    },

    // ...

    {
      "symbolicName": "DH2_State",
      "value": "CA"
    },
    {
      "symbolicName": "DH2_PropOne",
      "value": 7
    },
    {
      "symbolicName": "DH2_MVInt",
      "value": [
        0,
        101,
        200,
        210
      ]
    },
    {
      "symbolicName": "DH2_MVString",
      "value": [
        "One",
        "Three",
        "Thirty"
      ]
    },
    {
      "symbolicName": "DH2_City",
      "value": "San Diego"
    }
  ]
}
```

4. The external data service responds with values for the properties that it manages:

```
{
  "externalDataIdentifier": "1,1",
  "properties": [
    {
      "symbolicName": "DH2_State",
      "required": true,
      "maxLength": 2,
      "hasDependentProperties": true,
      "choiceList": {
        "displayName": "StateChoiceList",
        "choices": [
          {
            "displayName": "New York",
            "value": "NY"
          },
          {
            "displayName": "California",
            "value": "CA"
          },
          {
            "displayName": "Nevada",
            "value": "NV"
          }
        ]
      }
    },
    {
      "symbolicName": "DH2_PropOne",
      "maxValue": 10,
      "minValue": 1,
      "hasDependentProperties": false
    },
    {
      "symbolicName": "DH2_MVInt",
      "maxValue": 1000,
      "minValue": 0,
      "hasDependentProperties": true
    },
    {
      "symbolicName": "DH2_MVString",
      "required": true,
      "maxLength": 24,
      "hasDependentProperties": false,
        "choiceList": {
          "displayName": "MVStringChoiceList",
          "choices": [
            {
              "displayName": "One",
              "value": "One"
            },
            {
              "displayName": "Two",
              "value": "Two"
            },
            {
              "displayName": "Three",
              "value": "Three"
            },
            {
              "displayName": "Twenty",
              "value": "Twenty"
            },
            {
              "displayName": "Thirty",
              "value": "Thirty"
            },
            {
              "displayName": "Fourty",
              "value": "Fourty"
            }
          ]
        }
      },
      {
        "symbolicName": "DH2_City",
        "hidden": false,
        "required": true,
        "hasDependentProperties": false,
        "choiceList": {
          "displayName": "CityChoiceList",
          "choices": [
            {
              "displayName": "Los Angeles",
              "value": "Los Angeles"
            },
            {
              "displayName": "San Diego",
              "value": "San Diego"
            },
            {
              "displayName": "San Francisco",
              "value": "San Francisco"
            }
          ]
        }
      }
    ]
  }
```

5. The IBM Case Manager REST protocol validates the values that are submitted by the Case Manager Client with the final external data constraints. If no errors occur, the protocol creates the case and returns details for the new case:

```
{
  "CaseTitle": "DH2_MyCase_000000100602",
  "CaseIdentifier": "DH2_MyCase_000000100602",
  "CaseFolderId": "{7F390468-7FAD-43EB-B373-675D2255BB61}"
}
```

Content Platform Engine add-on extensions for IBM Case Manager

IBM Case Manager includes Content Platform Engine add-on extensions that are used by Case Manager Builder and Case Manager Client. These add-on extensions are modules that contain custom metadata and data that is stored in the design and target object stores. The custom metadata includes classes that are derived from base Content Engine classes. These add-on extensions provide the core object model, history, and analytics support for IBM Case Manager.

Using the Content Engine APIs, you can extend some of the custom classes to develop a customized solution. These custom classes include the Case Folder and Case Task classes, which are basic components of an IBM Case Manager solution. Both the Case Folder and Case Task classes are abstract classes and must be subclassed for a solution. Other classes, such as Deployed Solution, Deployed Case Type, and Case Comment, can be used as-is or can be subclassed. All of these custom classes are enabled to support case history and analytics.

As an example, you might want to extend the Case Folder class to add application-specific metadata. The base Case Folder class includes a few properties that reflect the type of case, the case state, the case ID, and the document that initiated the case. A subclass of the Case Folder class might be added to represent an insurance policy application. To this subclass, you might add properties for the name of the applicant, contact information, date of birth, requested policy amount, or other information. Some of the operations you can perform in a customized solution include:

- Querying for cases
- Creating a case
- Updating case properties
- Retrieving the metadata for case classes and document classes
- Adding a folder in a case folder
- Adding a document to a case

"IBM Case Manager design object store extensions"
"IBM Case Manager target object store extensions" on page 130
"IBM Case Manager history and analytics extensions" on page 135
"IBM Case Manager subscriptions and events" on page 136

IBM Case Manager design object store extensions

The IBM Case Manager design object store extensions include metadata that is required for IBM Case Manager design object store functions. The extensions provide property templates and implement custom classes, instances, and properties.

You can use IBM Administration Console for Content Platform Engine to view the metadata. You can also view the following properties for the design object store AddOn:

Display name
<*Release*>Case Manager Design Object Store Extensions.

XML import data
CM_CMDOS_CEExtensions.xml

Installation type
Optional.

Prerequisites
Base Content Engine Extensions.

Required by
None.

Important: Do not modify the values for properties on object classes that are created by IBM Case Manager. Changing these values can cause application behavior issues. In addition, do not extend the classes for other software applications.

Choice lists

The IBM Case Manager design object store extensions define choice lists that provide collections of predefined property values that you can use to present users with a list of values from which to choose. To review these choice lists in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the design object store.
2. In the object store navigation pane, click **Data Design** > **Choice Lists**.

The following table lists the choice lists that are defined by the IBM Case Manager design object store extensions:

Display name	Description
CmAcmIntegrationTypeChoiceList	Defines choices for the types of repository in which case documents are stored. By default, this value is set to FileNet P8 repositories for solutions, project areas, target environments, and target object stores. You can use the Configure IBM Content Manager Host Properties task in the IBM Case Manager configuration tool to set this value to IBM Content Manager repositories.
CmAcmRuleTypeChoiceList	Defines choices that indicate whether a business rule is text-based or table-based.
CmAcmTargetEnvironmentTypeChoiceList	Defines choices that indicate whether the target environment is a development environment or a production environment.
CmAcmTypeChoiceList	Defines choices that indicate the type of a page. The choice list is used by the Type (CmAcmType) property of the Page (CmAcmPage) class.
CmAcmVcsStatusChoiceList	Defines the choices that indicate the status of the Commit and Deliver actions in Case Manager Builder, which are related to version control system (VCS) integration. The choice list is used by the Commit Status (CmAcmVcsCommitStatus) and Deliver Status (CmAcmVcsDeliverStatus) properties of the VCS Execution State (CmAcmVcsExecutionState).

Custom object classes

The IBM Case Manager design object store extensions define custom objects that are used to track solution artifact definitions and solution locks. To review these custom objects in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the design object store.
2. In the object store navigation pane, click **Data Design** > **Classes** > **Custom Object**.

The following table lists the custom objects that are defined by the IBM Case Manager design object store extensions:

Display name (symbolic name)	Description
Draft Area (CmAcmDraftArea)	Defines an area that contains the saved solution artifact definitions.
Solution Lock Control (CmAcmSolutionLockControl)	Defines the definition entries for locks in a solution and tracks the save sequence number for the solution.

Document classes

The IBM Case Manager design object store extensions define classes that represent various document objects that are associated with a solution. To review these document classes in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the design object store.
2. In the object store navigation pane, click **Data Design** > **Classes** > **Document**.

The following table lists the document classes that are defined by the IBM Case Manager design object store extensions:

Display name (symbolic name)	Description
Connection Definition (CmAcmConnectionDefinition)	Defines the Case Manager Builder or Case Manager Client association with the target environment. One connection definition exists for each target environment to which a case management solution is deployed.
Page (CmAcmPage)	Defines an IBM Case Manager page, which contains the widgets that are required to complete a task.
Rule (CmAcmRule)	Defines a business rule in IBM Case Manager.
VCS Execution State (CmAcmVcsExecutionState)	Defines the status of actions that are related to the version control system for a solution, such as commit and deliver.
View (CmAcmView)	Defines a view for the Properties widget.
Widgets Package (CmAcmWidgetsPackage)	Defines a package of widgets that can be used with IBM Case Manager.

Folder classes

The IBM Case Manager design object store extensions define classes that represent various folder objects that are used as containers for solution objects. To review these document classes in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the design object store.

2. In the object store navigation pane, click **Data Design** > **Classes** > **Folder**.

The following table lists the folder classes that are defined by the IBM Case Manager design object store extensions:

Display name (symbolic name)	**Description**
ReinitArtifacts (CmAcmReinitArtifacts)	Defines a folder that contains artifacts that are used when the test environment is reset. These artifacts include code modules and documents from the target object store, the manifest of the target environment, and the status of the most recent (current or past) reset of the test environment.
Solution Folder (CmAcmSolutionFolder)	Defines a folder for a solution.

Related concepts:

Administering Content Platform Engine

Base Content Engine Extensions

IBM Case Manager target object store extensions

The IBM Case Manager target object store extensions provide metadata that is required for IBM Case Manager target object store functions. The extensions provide property templates and implements custom classes, instances, and properties.

You can use IBM Administration Console for Content Platform Engine to view the following metadata:

Display name
: <*Release*> Case Manager Target Object Store Extensions.

XML import data
: `CM_CMTOS_CEExtensions.xml`

Installation type
: Optional.

Prerequisites
: Base Content Engine Extensions.

Required by
: IBM Case Manager history and analytics extensions.

Choice lists

The IBM Case Manager target object store extensions define choice lists that provide collections of predefined property values that you can use to present users with a list of values from which to choose. To review these choice lists in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the target object store.
2. In the object store navigation pane, click **Data Design** > **Choice Lists**.

The following table lists the folder classes that are defined by the IBM Case Manager target object store extensions:

Display name	Description
CmAcmActionChoiceList	Defines choices that indicate the type of a comment.
CmAcmAuditConfigurationStateChoiceList	Defines choices that indicate the status of an applied audit configuration.
CmAcmCaseStateChoiceList	Defines choices that indicate that state of a case folder.
CmAcmDeploymentStateChoiceList	Defines choices that indicate the deployment state of a deployed solution folder.
CmAcmDisabledStateChoiceList	Defines choices that indicate the disabled state of a task.
CmAcmGroupModeChoiceList	Defines choices that indicate whether a task is not grouped, in an exclusive group, or in an inclusive group.
CmAcmIntegrationTypeChoiceList	Defines choices that indicate the type of repository with which IBM Case Manager is integrated.
CmAcmLaunchModeChoiceList	Defines choices that indicate the launch mode state of a task.
CmAcmPreconditionStateChoiceList	Defines choices that indicate the state of a task precondition.
CmAcmRelationshipTypeChoiceList	Defines choices that indicate the type of relationship between two cases.
CmAcmRequiredStateChoiceList	Choice items for the possible required states for a case task.
CmAcmSecurityConfigurationStateChoiceList	Defines choices that indicate the status of an applied security configuration.
CmAcmTargetEnvironmentTypeChoiceList	Defines choices that indicate whether the target environment is a development environment or a production environment.
CmAcmTriggerTypeChoiceList	Defines choices that indicate the trigger type for a task type.
CmAcmUserLaunchedTaskWorkflowTypeChoiceList	Defines choices that indicate whether a workflow is a Content Platform Engine workflow or an IBM Business Process Manager workflow.
IcnRepositoryTypeChoiceList	Defines choices that indicate the type of IBM Content Navigator repository that is used for an external document.

Custom object classes

The IBM Case Manager target object store extensions define custom objects that are used to track task types, precondition parameters, and workflow parameters. To review these custom objects in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the target object store.
2. In the object store navigation pane, click **Data Design** > **Classes** > **Custom Object**.

The following table lists the custom objects that are defined by the IBM Case Manager target object store extensions:

Display name (symbolic name)	Description
Deployed Task Type Info (CmAcmDeployedTaskTypeInfo)	Represents a task type that is deployed for a solution.

Display name (symbolic name)	Description
Precondition Checker Parameters (CmAcmPreconditionCheckerParameters)	Tracks the options and filters that are passed as parameter values for the precondition checker.
Task Workflow Parameters (CmAcmTaskWorkflowParameters)	Represents the mapping of the parameters that are used in a task workflow.

Document class

The IBM Case Manager target object store extensions define a class that represents an external document that is associated with a case. To review this document class in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the target object store.
2. In the object store navigation pane, click **Data Design** > **Classes** > **Document**.

Display name (symbolic name)	Description
External Document (IcnExternalDocument)	Represents a document that is stored in a repository other than this case management target object store.

Folder classes

The IBM Case Manager target object store extensions define classes that represent various folder objects that are used as containers for solution and case objects. To review these document classes in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the target object store.
2. In the object store navigation pane, click **Data Design** > **Classes** > **Folder**.
3. For certain classes, expand the parent class as indicated in the following table.

The following table lists the folder classes that are defined by the IBM Case Manager target object store extensions:

Display name (symbolic name)	Description
Base Case (CmAcmBaseCase)	Represents the base abstract class for the Case Folder and Case Subfolder classes.
Case Folder (CmAcmCaseFolder)	Represents the base, abstract class for Case Folder instances. The Case Folder class is the class from which subclasses for case types that are part of a solution are derived. To view this class, click **Base Class** in the object store navigation pane.
Case Subfolder (CmAcmCaseSubfolder)	Represents a subfolder under a Case Folder instance. To view this class, click **Base Class** in the object store navigation pane.
Case Type Subfolder (CmAcmCaseTypeSubfolder)	Represents a folder that is used to enable security inheritance from a Deployed Case Type folder down to a Case folder.
Deployed Case Type (CmAcmDeployedCaseType)	Represents a Case Type instance within a Deployed Solution. Certain artifacts of that case type are in this folder hierarchy.

Display name (symbolic name)	Description
Deployed Solution (CmAcmDeployedSolution)	Represents a Case Solution folder in a deployed solution. Certain solution artifacts and the case types and instances are in this folder hierarchy.

Other classes

The IBM Case Manager target object store extensions define other classes that represent different types of comments and tasks, subscriptions, and case relationships. To review these other classes in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the target object store.
2. In the object store navigation pane, click **Data Design** > **Classes** > **Other Classes**.
3. For certain classes, expand the parent class as indicated in the following table.

Display name (symbolic name)	Description
Case Comment (CmAcmCaseComment)	Represents the base class for comments that are associated with a specific case. A case comment can be associated with the case or with a document, task, or work item in the case. To view this class, click **Annotation** in the object store navigation pane.
Case File Event (CmAcmCaseFileEvent)	Records the filing of a document into a case for auditing purposes. To view this class, click **Event** > **Object Change Event** > **Custom Event** in the object store navigation pane.
Case Related Event (CmAcmCaseRelatedEvent)	Records the creation of a relationship between two cases for auditing purposes. To view this class, click **Event** > **Object Change Event** > **Custom Event** in the object store navigation pane.
Case Relationship (CmAcmCaseRelationship)	Represents the relationship between two cases. To view this class, click **Link** in the object store navigation pane.
Case Task (CmAcmCaseTask)	Represents the base, abstract class for a Case Task. To view this class, click **Task** in the object store navigation pane.
Case Unfile Event (CmAcmCaseUnfileEvent)	Records the unfiling of a document from a case for auditing purposes. To view this class, click **Event** > **Object Change Event** > **Custom Event** in the object store navigation pane.
Directory Validation Event (CmAcmDirectoryValidationEvent)	Validates the presence of the rules repository directory on the Content Platform Engine server. To view this class, click **Event** > **Object Change Event** > **Custom Event** in the object store navigation pane.

Display name (symbolic name)	Description
Document Create Case Subscription (CmAcmDocumentCreateCaseSubscription)	Represents a synchronous subscription to the Create event on Document classes. Class subscriptions of this class are created by the solution deployment tool. To view this class, click **Class Subscription** in the object store navigation pane.
Dynamic Task (CmAcmDynamicTask)	Represents a custom task that a user creates in Case Manager Client. To view this class, click **Task** > **Case Task** in the object store navigation pane.
Rule Deployment Event (CmAcmRuleDeploymentEvent)	Causes rules to be deployed to the rules repository directory on the Content Platform Engine server. To view this class, click **Event** > **Object Change Event** > **Custom Event** in the object store navigation pane.
Rule Export Event (CmAcmRuleExportEvent)	Causes the rules in a solution to be exported to a package and uploaded to the design object store To view this class, click **Event** > **Object Change Event** > **Custom Event** in the object store navigation pane.
Rule Reset Event (CmAcmRuleResetEvent)	Causes the rules repository directory to be reset. To view this class, click **Event** > **Object Change Event** > **Custom Event** in the object store navigation pane.
Task Comment (CmAcmTaskComment)	Represents a comment for a case task. To view this class, click **Annotation** > **Case Comment** in the object store navigation pane.
Task With Initiating Document (CmAcmTaskWithInitiatingDocument)	Represents the base, abstract class for a Case Task that has a file precondition. To view this class, click **Task** > **Case Task** in the object store navigation pane.
Task Workflow Launch Subscription (CmAcmTaskWorkflowLaunchSubscription)	Represents an asynchronous subscription to the Change State event of a Case Task. To view this class, click **Class Subscription** in the object store navigation pane.
Version Series Comment (CmAcmVersionSeriesComment)	Represents a comment for a document in a case. To view this class, click **Annotation** > **Case Comment** in the object store navigation pane.
Work Item Comment (CmAcmWorkItemComment)	Represents a comment for a case work item. To view this class, click **Annotation** > **Case Comment** > **Task Comment** in the object store navigation pane.

Related concepts:

Administering Content Platform Engine

Base Content Engine Extensions

Related reference:

"IBM Case Manager history and analytics extensions" on page 135

IBM Case Manager history and analytics extensions

The IBM Case Manager history and analytics extensions include metadata that configures Content Platform Engine for auditing of analytical and historical information by IBM Case Manager client applications.

The IBM Case Manager history and analytics extensions consist of an event class and event class properties, and is deployed to the IBM Case Manager target object store. You can use IBM Administration Console for Content Platform Engine to view the metadata.

Display Name
: *<Release>* Case Manager History and Analytics Extensions.

XML Import Data
: `CMHistoryAndAnalyticsAddOn.xml`.

Installation Type
: Optional.

Prerequisites
: IBM Case Manager target object store extensions.

Required By
: None.

The IBM Case Manager history and analytics extensions configure several properties on source object classes to be audited individually. Only the individual properties are then recorded, rather than the entire source object on the audited event. For a property to be audited on a source object, the corresponding property definition on the class must be configured for auditing. Specifically, the `AuditAs` property of the property definition must be set to an event property that holds the value of the source object property.

The classes contain properties that are audited by default and where their corresponding values are found on the event objects.

For a property that is not audited by default, you can set the `AuditAs` property in the property definition to audit that property. When an event is raised on the object, the property is then audited.

To set the `AuditAs` property in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the target object store.
2. In the object store navigation pane, click **Data Design** > **Classes** and then click the class.
3. On the Properties Definition tab, click the property that you want to audit to open the Property Definition dialog box. Click the More tab.
4. From the **Audit as** list, select the event property template that is to be used to audit the property.

The following table lists the auditable classes that are defined by the IBM Case Manager target object store extensions:

Display name (symbolic name)	Description
Case Comment (CmAcmCaseComment)	Represents the base class for comments that are associated with a specific case. A case comment can be associated with the case or with a document, task, or work item in the case.
Case Folder (CmAcmCaseFolder)	The Case Folder class and its subclasses are extended with auditing configuration settings that are accessible by the applications that you configure for case analytics and for case history.
Case Subfolder (CmAcmCaseSubfolder)	Represents a subfolder under a Case Folder instance.
Case Task (CmAcmCaseTask)	Represents the base, abstract class for a Case Task.
Document (Document)	A single version of a document stored in an object store.
Task Comment (CmAcmTaskComment)	Represents a comment for a case task.
Version Series Comment (CmAcmVersionSeriesComment)	Represents a comment for a document in a case.
Work Item Comment (CmAcmWorkItemComment)	Represents a comment for a case work item.

Related concepts:

Administering Content Platform Engine

Content Engine APIs: property auditing

Related reference:

"IBM Case Manager target object store extensions" on page 130

IBM Case Manager subscriptions and events

The IBM Case Manager subscriptions, event actions, and code module are deployed to the IBM Case Manager target object store. You can use IBM Administration Console for Content Platform Engine to view the metadata.

Important: Although in IBM Administration Console for Content Platform Engine, you can change events from synchronous to asynchronous or from asynchronous to synchronous, doing so for IBM Case Manager events can cause unintended behavior.

The following table lists the event and subscription classes that are defined by the IBM Case Manager target object store extensions. To review these choice lists in IBM Administration Console for Content Platform Engine:

1. In the domain navigation pane, select the target object store.
2. In the object store navigation pane, click **Data Design** > **Classes** > **Other Classes**. Then, to view an event class, click **Event** > **Object Change Event** > **Custom Event**. To view a subscription class, click **Class Subscription**.

Display name (symbolic name)	Description
Case File Event (CmAcmCaseFileEvent)	Represents an event that occurs when a Case Folder or Case Subfolder has its file method called to file a Document object.
Case Related Event (CmAcmCaseRelatedEvent)	Represents an event that occurs when a relationship is established between two cases.
Case Unfile Event (CmAcmCaseUnfileEvent)	Represents an event that occurs when a document is unfiled from a case.

Display name (symbolic name)	Description
Directory Validation Event (CmAcmDirectoryValidationEvent)	Represents an event that occurs when IBM Case Manager must verify that the Rules repository directory is present on the Content Platform Engine server.
Document Create Case Subscription (CmAcmDocumentCreateCaseSubscription)	Represents a synchronous subscription to the Create event on Document classes. Class subscriptions of this class are created by the solution deployment tool.
Rule Deployment Event (CmAcmRuleDeploymentEvent)	Represents an event that occurs when a business rule is deployed.
Rule Export Event (CmAcmRuleExportEvent)	Represents an event that occurs when a business rule is exported.
Rule Reset Event (CmAcmRuleResetEvent)	Represents an event that occurs when a business rule is reset.
Task Workflow Launch Subscription (CmAcmTaskWorkflowLaunchSubscription)	Represents an asynchronous subscription to the Change State event of a Case Task.

Related concepts:

Administering Content Platform Engine

Subscribable and Auditable Events

Content Engine APIs: subscription concepts

Using external properties

External properties are ad-hoc properties that are not defined within the properties of a case or task, but that can be rendered within their associated views in IBM Case Manager, for example, the Properties widget. External properties are defined, retrieved, and persisted by using an external data source such as a JAVA servlet, JSON file, or data service. The binding of external properties is handled by the Properties area in the model layer.

General use case: Most organizations maintain several Systems of Record data sources. These data sources often contain the most current single source of truth. Easy access to this information helps case workers complete their goals. It is often necessary and efficient to access this data live and not store a copy.

Scenario: While working a customer claim, an agent opens a case to view the details of the case. In addition to claim-specific data, the agent can see the customer's contact information, which is pulled live from the customer database. A list of past transactions with the customer is also conveniently displayed, which saves the agent from having to access the transaction system for this information.

"Defining external properties at run time"

"Defining external properties by using the Script Adapter widget" on page 140

"Retrieving and persisting external properties" on page 143

Related information:

Adding external properties to the properties view

Defining external properties at run time

This simple and quick example demonstrates the extensibility, flexibility, and portability of external properties.

About this task

Procedure

To define and bind an external property at run time:

1. Log on to Case Manager Builder.
2. From the Add Case page or the Case Details page, add a Script Adapter widget to the page.
3. On the Script Adapter widget:
   a. Click the **Edit Wiring** icon and wire the widget to receive 'Send new case information' events.
   b. Click the **Edit Settings** icon and paste the following JavaScript in the window. This JavaScript defines an external property called PhoneNumber.

```
require([
 "icm/model/properties/controller/ControllerManager",
 "icm/base/Constants"], function(ControllerManager, Constants) {
    // Get the editable and coordination objects from the event payload.
    var coordination = payload.coordination;
    var editable = payload.caseEditable;
    var model;
    // Participate in the BEFORELOADWIDGET topic to bind the external
    // properties into the controller.
    payload.coordination.participate(Constants.CoordTopic.BEFORELOADWIDGET,
         function(context, complete, abort) {
       model = {
          properties: {
             "PhoneNumber": {
                 id: "PhoneNumber",
                 name: "Phone Number",
                 type: "string",
                 cardinality: "single",
                 value: "949-559-2213"
             }
          }
     };
     var collectionController = ControllerManager.bind(editable);
     collectionController.bind("External", "External", model);
     complete();
 });

 // Participate in the AFTERLOADWIDGET topic to release
 // the controller binding.
 payload.coordination.participate(Constants.CoordTopic.AFTERLOADWIDGET,
    function(context, complete, abort) {
       ControllerManager.unbind(editable);
       complete();
 });
});
```

Important: All external property-related binding typically should happen in the handler for the BEFORELOADWIDGET.

4. Save the page, then save and deploy your changes to the solution.
5. Open the solution in Case Client and add a case. Notice that the external property, Phone Number, appears with the case properties on the page for the new case.

Defining external properties by using the Script Adapter widget

This example shows how to define external properties by using JavaScript in a Script Adapter widget for the Properties widget in Case Client.

About this task

Procedure

To define external properties by using the Script Adapter widget:

1. Define an external properties collection in the model layer as a single object.

   For example, provide a collection of property objects:

```
var model = {
    properties: {
        "name": {
            id: "name",
            type: "string",
            value: "Rip Van Winkle"
        },
        "age": {
            id: "age",
            type: "integer",
            value: 200
        },
    },
}
```

2. Implement the definition, retrieval, and persistence of external properties by using custom code. Typically, this custom code is implemented in the Script Adapter widget.

   Your custom code can call the bind method of the PropertyCollectionController class, as shown below, to include your external properties collection in the set of properties that are associated with the view. You can bind as many collections as you wish. Specify a unique collection identifier for each collection.

   ```
   controller.bind(collectionId, collectionName, model);
   ```

   The PropertyCollectionController class supports standard model signatures with which it can bind implicitly. In some cases, your application might need to support a non-standard model signature that is associated with your application. If so, you can provide an Integration object as part of the binding to instruct the controller how to interact with the associated model.

   ```
   controller.bind(collectionId, collectionName, model, integration);
   ```

   The PropertyCollectionController class automatically updates the state of the model as changes occur within the view.

3. Set up wiring for the Script Adapter widget.

   It is important to specify an incoming event for the Script Adapter widget that is loaded before the actual widget loads on the page. For example, for the Add Cases page, wire the Script Adapter widget to the Page Container's 'Send new case information' event. Similarly, you can use the Page Container's 'Send case information event' for the Case Details page and the Page Container's 'Send work item' event for the Work Details page. These events allow the Script Adapter widget to execute the JavaScript code so that the external properties are bound to the Properties widget before it is loaded.

4. Use external properties that are defined by using a model object.

   The following example illustrates the definition of a typical collection of external properties of various data types.

   ```
   {
       properties: {
           "description": {
               id: "description",
               name: "Description",
               label: "Description",
               type: "string",
               cardinality: "single",
               value: "description here"
           },
           "price": {
               id: "price",
               name: "Price",
               label: "Price",
               type: "float",
               cardinality: "single",
               value: 22.2
           },
           "booleantest": {
               id: "booleantest",
               name: "booleantest",
               label: "booleantest",
               type: "boolean",
               cardinality: "single"
           },
   ```

```
        "datetimeTEST": {
            id: "datetimeTEST",
            name: "datetimeTEST",
            label: "datetimeTEST",
            type: "datetime",
            cardinality: "single"
        },
        "quantityINT": {
            id: "quantityINT",
            name: "quantityINT",
            label: "quantityINT",
            type: "integer",
            cardinality: "single"
        },
        "total": {
            id: "total",
            name: "Total",
            label: "Total",
            type: "float",
            cardinality: "single",
        },
        "MyMultiInteger": {
            id: "MyMultiInteger",
            type: "integer",
            cardinality: "multi",
            value: [1, 2, 3]
        },
        "multiCategory": {
            id: "multiCategory",
            name: "MultiCategory",
            label: "MultiCategory",
            type: "integer",
            cardinality: "multi",
            choices: [{
                    label: "Small",
                    value: 0
                },
                {
                    label: "Large",
                    value: 1
                }
            ]
        }
    }
}
```

5. Create an Integration object.

   Creating a custom Integration object is useful when you are working with a third-party model object with a different signature than the default signature that is supported by the controller. Some examples include an incoming JSON object from a web service or an object from within your existing model layer. You can create a custom Integration object that tells the controller how to map its attributes to fields of the custom object.

   The following script demonstrates how to create the Integration object that is used for such a binding. The bold below shows the parts that are important to note for custom integration configuration.

```
// Create the external properties model with the custom model signature.
var model = {
    props: {
        "PhoneNumber": {
            symbolicName: "PhoneNumber",
            name: "Phone Number",
            type: "string",
            multiValue: false,
            value: "949-559-2213"
        }
    }
};

// Create a custom integration object for the custom model signature.
var integration = new Integration();
integration.mergeConfiguration(basicIntegrationConfiguration);
integration.mergeConfiguration(customIntegrationConfiguration);

// Add a binding for the external properties to the controller.
collectionController.bind("External", "External", model, integration);
```

The following script illustrates the custom integration configuration object that is required for the custom integration in the previous script. Typically, this configuration object is implemented in a separate Dojo module. Merge the basic integration configuration before you merge your custom integration configuration as shown in the previous script.

```
var customIntegrationConfiguration = {
    bindings: {
        collection: {
            attributes: {
                properties: {
                //Get the properties from the "props" member of the model.
                    get: "props"
                        }
            }
        },
        property: {
            attributes: {
                common: {
                    id: {
                         //Get the id from the "symbolicName" member of
                         //the model object.
                         get: "symbolicName"
                    },
                    cardinality:
                          // Compute the cardinality from the "multiValue"
                          //member of the model object.
                          get: function(model) {
                              return model.multiValue ? "multi" : "single";
                    }

                }
            }
        }
    }
}
```

Retrieving and persisting external properties

The retrieval and persistence of external properties depends entirely on your unique application requirements. It must therefore be implemented in custom code.

In one scenario, external properties might be located in a database. In another scenario, they might be from a web service. A third scenario might simply use the data to update the visible status of a widget.

Typically, you coordinate the retrieval of external properties with the rendering of a page and you coordinate the persistence of external properties with the persistence of a page. A Coordination object is provided to organize these activities among the various widgets on a page. Your custom code in the Script Adapter widget can participate in this coordination as demonstrated by the code that follows. Note that getExternalProperties and setExternalProperties are application-specific methods that are provided in your custom code.

```
require(["icm/base/Constants", "icm/model/properties/controller/ControllerManager"],
   function(Constants, ControllerManager) {
      /* Get the coordination and editable objects from the event payload. */
      var coordination = payload.coordination;
      var editable = payload.caseEditable;
      /* Use the BEFORELOADWIDGET coordination topic handler to obtain the
           controller binding for the editable and to update the properties. */
      coordination.participate(Constants.CoordTopic.BEFORELOADWIDGET,
         function(context, complete, abort) {
            /* Obtain the controller binding for the editable. */
            var controller = ControllerManager.bind(editable);
            /* Retrieve the external properties and bind them to the controller. */
            var externalProperties = getExternalProperties();
            /* You must provide this function. */
            controller.bind("Ext1", "Ext1", externalProperties);
            /* You can optionally provide an integration object if a
               non-standard model is used. */
            /* Call the coordination completion method. */
            complete();
         });
   /* Use the SAVE coordination topic handler to release the controller binding
      for the editable. */
   coordination.participate(Constants.CoordTopic.SAVE,
         function(context, complete, abort) {
      /* Release the controller binding for the editable. */
      ControllerManager.unbind(editable);
      /* Will automatically release the external properties binding. */
      /* Save the external properties. */
      saveExternalProperties(externalProperties);
      /* You must provide this function. */
      /* Call the coordination completion method. */
      complete();
   });
});
```

Creating custom property editors and controllers

You can create custom editors and controls to use with properties views. For example, you might create a custom editor and controller to display a record from an external data source such as a customer relationship management system.

About this task

To use your custom editors and controllers, you must create and add an extensions package to your IBM Case Manager environment. The following procedure provides an overview of the steps that are required to create a custom property editor and controller and the extensions package. For detailed instructions on creating the editor and controller, see Creating custom property editors and controllers in IBM Case Manager V5.2.1 on the IBM developerWorks® website.

Procedure

To create an extensions package for a custom property editor and controller:

1. Create a web project that contains the following folders for your extensions package:

Folder	Content
ProjectName/*ProjectName*Plugin	Contains the files that are used to create the JAR file for the IBM Content Navigator plug-in.
ProjectName/*ProjectName*Plugin/src/*PackageName*	Contains the files that are used to create the JAR file for the IBM Content Navigator plug-in.
ProjectName/*ProjectName*Plugin/src/*PackageName*/WebContent	Contains the main JavaScript plug-in file and the root folder of your custom editors and controller code. It can have subfolder structures to organize the code packages.
ProjectName/`ICMRegistry`	Contains the `Extension.json` file that are used to register the extensions package and optionally the translated `Extension.json` in the `nls` subfolder.

2. Create the registry files and place them in the `ICMRegistry` folder:
   a. Create a file called `Extension.json`. This JSON-format file indicates the ID, title, description, type, packages, CSS, and a bootstrap class of the extensions.
   b. Optional: For a different locale, you can create the translated `Extension.json` files, and put them in the corresponding language folders under the `nls` subfolder. For example, create `ICMRegistry/nls/fr/Extension.json` for a French locale.
3. Create a standard Content Navigator plug-in in the `ProjectName/ProjectNamePlugin` folder, to hold the source code for your custom editors and controllers. Create the following items in the `WebContent` folder of the plug-in:
   a. Create a self-contained Dojo widget to represent the customer editor that you want to use in the properties view.

   b. Create a registry file to describe the custom editor, and specify the types of the properties that are suitable to use with the editor in the registry. This registry file is used to register the editor into Properties View Designer.
   c. Optional: If you want to interact with custom data types, you can create a custom controller to use with the editor and the custom data type. You must also create a custom integration configuration file to ingest the custom controller in the integration configuration.
   d. Create a bootstrap class to register the custom editor and custom controllers.
4. Create a extensions package that contains the custom the custom plug-in and registration file:
   a. Create a `build.xml` script that builds the following components:
      - The `ICMRegistry` folder that includes the extension definitions
      - A JAR file that contains the IBM Content Navigator plug-in
5. In the IBM Case Manager configuration tool, run the Deploy and Register Extensions Package task to register and deploy your extensions package.

   Important: If you run this task in a cluster environment, you must ensure that the plug-in is loaded on each node of the cluster. Either restart the cluster to force the plug-in to be loaded on all nodes or manually load the plug-in on each node by using the IBM Content Navigator administration client.
6. In Case Manager Builder, use Properties View Designer to choose the custom properties editor for a property in a properties view.
7. Deploy and test your solution.

Creating custom inline messages and prompts

When a text-box field, such as a Number Text Box Editor, is empty, Case Manager Client displays a prompt message as a popup tooltip. After the user enters data in the field, the tooltip goes away. In addition, Case Manager Client displays default messages if the user enters a value that is invalid or outside the range for the property.

You can provide custom prompts and messages by creating a custom JavaScript that uses the set method for the Controller class as shown in the following example:

```
var propertyController =
 controller.getPropertyController("F_CaseFolder", "ABC_Property1");
propertyController.set('promptMessage", "Enter your favorite color");
propertyController.set("invalidMessage",
 "The value that you entered is not valid.");
propertyController.set("rangeMessage", "Enter a value between {0} and {1}");
```

Creating custom page widgets and actions

You can create custom page widgets to use with or in place of the IBM Case Manager page widgets. For example, you might create a custom widget to display a record from an external data source such as a customer relationship management system. You might create a widget that replaces the IBM Case Manager Search widget with a user interface that customizes the display of search properties for your users.

Before you begin

Case Manager Client and the page widgets run in IBM Content Navigator. Therefore, before you create custom widgets, you must set up your development environment to customize and extend IBM Content Navigator. For information, see section 5.1, "Preparing for IBM Content Navigator customization," in the IBM Redbooks® publication Customizing and Extending IBM Content Navigator.

About this task

The following procedure provides an overview of the steps that are required to create a custom page widget. For detailed instructions and samples, see Creating custom widgets with the IBM Case Manager JavaScript API on the IBM developerWorks website.

Procedure

To create a custom page widget:

1. Create a web project that contains the following folders for your widget package:

Folder	Content
ProjectName/*ProjectName*Plugin	Contains the files that are used to create the JAR file for the IBM Content Navigator plug-in.
ProjectName/*ProjectName*Plugin/src/*PackageName*	Contains the files that are used to create the JAR file for the IBM Content Navigator plug-in.
ProjectName/*ProjectName*Plugin/src/*PackageName*/WebContent	Contains the CSS files, the main JavaScript plug-in file and related files such as images.
ProjectName/`ICMRegistry`	Contains the JSON files that are used to register the widget package and the page widgets. Optionally, this folder can contain folders for: • Images that are used in the widget package, such as icons or thumbnails • Translated resource files
ProjectName/*ProjectName*Widget	Contains the files that are used to define the user interface for a custom page widget.
ProjectName/*ProjectName*Widget.*PackageName*/pgwidget	Contains the files that are used to define the custom page widgets.

Folder	Content
ProjectName/ *ProjectName*Widget.*PackageName*/action	Contains the files that are used to define the custom actions that are used by the custom page widgets.

For more information, see section 5.2.1, “General structure of a plug-in project,” in the IBMRedbooks publication Customizing and Extending IBM Content Navigator.

2. Create the registry files and place them in the `ICMRegistry` folder:
   a. Create a file called `Catalog.JSON`. This JSON-format file identifies the widget category and the page widgets that the package contains.
   b. For each page widget, create a definition file in JSON format that identifies the properties, toolbars, menus, and actions that can be configured for the widget.
3. Create a self-contained widget that is based on Dojo to represent the user interface component of the custom page widget.

   Do not include the business logic for the page widget in this file. Instead, use this file to define the visual representation of the widget that is displayed to users in Case Manager Client. In addition, include a `destroy` method to be called to close the widget when the page that contains the widget is closed.
4. Create a wrapper file that defines a custom class to represent the page widget.

   You define the wrapper JS file in the Dojo Asynchronous Module Definition (AMD) format by calling the `Dojo.declare()` method.

   The class for the page widget must:
   - Extend the user interface component that you create in step 3.
   - Mix in the `icm.base.BasePageWidget` class. This class mixes in the `icm.base._EventStub` class that includes methods for publishing and broadcasting events.
   - If the page widget contains a toolbar or menu that uses the IBM Case Manager action framework, mix in the `icm/base/BaseActionContext` class.
   - If the page widget must interact with other page widgets to perform tasks such as adding or saving cases, tasks, or work items, participate in coordination.
5. If your page widget uses custom actions, define a class for each action.

   An IBM Case Manager action extends the `ecm.model.Action` class. To make a standard IBM Content Navigator action work in IBM Case Manager, the `com.ibm.ecm.extension.PluginAction` implementation must override the `getAdditionalConfiguration` method to provide the action definition.

   To define the class for a custom action, you extend the `icm.action.Action` class. You must implement an `execute` method in the class to define the operation logic for the action. Optionally, you can implement an `isEnabled` method and an `isVisible` method to check the state.

   Tip: You can customize the dialog boxes that are used to display error messages and confirmation messages for your custom actions. To override the default dialog boxes, use the `showConfirmationDialog` method and `showErrDialog` method that are defined for the `icm.action.Action` JavaScript class.
6. Create the IBM Content Navigator plug-in for the widget package. The plug-in contains the web browser logic that enables users to call the page widget.

For more information, see section 5.2.2, “Create a plug-in project from the samplePlugin code,” in the IBMRedbooks publication Customizing and Extending IBM Content Navigator.

7. Create a widget package that contains the custom page widgets and actions:
   a. Create a `build.xml` script that builds the following components:
      - An EAR file that contains the runtime code that implements the page widgets and actions
      - The `ICMRegistry` folder that includes the page widget definitions
      - A JAR file that contains the IBM Content Navigator plug-in, including the bootstrap file and the action definitions
   b. Create a zip file that contains the files and folder from the previous step.
8. Create a `MANIFEST.MF` file that is in the *`ProjectName`*`Plugin/src/META-INF` folder that contains t:he following reference to the *Custom plug-in*.js file:

   ```
   Plugin-Class: Custom plug-in
   ```
9. In the IBM Case Manager configuration tool, make and run a copy of the **Deploy and Register Widgets Package** task to register your widget package and to deploy it in your design environment.

   Important: If you run this task in a cluster environment, you must ensure that the plug-in is loaded on each node of the cluster. Either restart the cluster to force the plug-in to be loaded on all nodes or manually load the plug-in on each node by using the IBM Content Navigator administration client.

   The Deploy and Register Widgets Package task modifies only those components within the application server for IBM Content Navigator application server. For environments where client requests are routed through an HTTP server such as IBM HTTP Server, a load balancer, or so on, ensure that the endpoints are configured correctly. In addition, ensure that the HTTP server plug-ins are regenerated to allow clients access to the runtime code with the deployed EAR application.
10. In Case Manager Builder, use Page Designer to add the custom page widget to a page and configure the properties and actions for the page widget.
11. Deploy and test the solution.

“Defining registry files for custom actions, properties, page widgets, and events”

Related information:

Class icm.action.Action

Creating custom widgets with the IBM Case Manager JavaScript API

Defining registry files for custom actions, properties, page widgets, and events

You can include certain properties in the registry files for your custom action, properties, page widget, or event.

“Defining the widget package catalog file” on page 152

“Defining a page widget definition file” on page 156

“Defining an action definition file” on page 158

“Defining a property for a page widget or an action” on page 160

“Defining a property type” on page 161

"Defining a widget event" on page 163

Defining the widget package catalog file

The widget package catalog file is a JSON file that identifies the custom page widgets that are contained in your widget package. This file, which must be named `Catalog.JSON`, is in the `ICMRegistry` or `ICMRegistry/nls` folder of your widget package.

The following example shows the structure of the `Catalog.json` file.

```
{
    "Name":"IBM Case Manager Widget package",
    "Description":"Description of package",
    "Locale":"",
    "Version":"5.3.3",
    "Categories":[

        {
            "id":"EducationWidgets",
            "title":"Education Widgets"
        }
    ],
    "Widgets":[
        {

            "id":"CustomInbasket",
            "title":"Custom Inbasket",
            "category":"EducationWidgets",
            "description":"EN description of Custom Inbasket",
            "definition":"CustomInbasket.json",
            "preview":"images/custom/custominbasket_preview.gif",
            "icon":"images/custom/custominbasket_icon.gif",
            "runtimeClassName":"icm.pgwidget.inbasket.CustomInbasket",
            "previewThumbnail":"images/custom/custominbasket_thumb.gif"
        }
    ]
]
```

The following table describes the properties that are supported for the `Catalog.json` file.

Table 94. `Catalog.json` supported properties

Property	Required or Optional	Type	Description
Name	Required	String	A name for the custom page widget package. Specify a unique name for the package to avoid overriding an existing page widget package.
Description	Required	String	A description of the custom page widget package.

Table 94. `Catalog.json` supported properties (continued)

Property	Required or Optional	Type	Description
Locale	Required	String	The two-character locale code for the current catalog. For example, `zh` is the locale code for simple Chinese. The code is added as a subfolder name when the widget definition file is retrieved. By default, the locale is set to "".
Version	Optional	String	The version number that is assigned to the widget package.
Categories	Optional	String	The categories in Case Manager Builder in which the custom page widgets in this package are listed. You can choose to list the page widgets in one of the following categories, which are provided by IBM Case Manager: • *CaseWidgets* • *GenericWidgets* For each category, you must provide an identifier and title.
Categories/id	Required	String	A unique identifier for the widget category.
Categories/title	Required	String	The name that is to be displayed in Case Manager Builder for the widget category.

Table 94. `Catalog.json` supported properties (continued)

Property	Required or Optional	Type	Description
Widgets	Required	JSON array	An array that identifies the custom page widgets in this package. For each page widget, you must provide the following information: • id • category • title • description • definition • preview • icon • runtimeClassName • previewThumbnail
Widgets/id	Required	String	A unique identifier for the page widget.
Widgets/category	Required	String	The identifier of the category in which the page widget is to be listed in Case Manager Builder.
Widgets/title	Required	String	The name to be displayed for the page widget in Case Manager Builder.
Widgets/description	Required	String	A description of the page widget. This text is used as hover help for the widget in Case Manager Builder.
Widgets/definition	Required	String	The full path and name of the definition file for the page widget.

Table 94. `Catalog.json` supported properties (continued)

Property	Required or Optional	Type	Description
Widgets/preview	Required	String	The relative path and name of the resource file that contains the preview image for the page widget. For example, the value might be `images/myWidget_prv.png`. The image can be a .png file or a .gif file. This image is not used in IBM Case Manager V5.2.
Widgets/icon	Required	String	The relative path and name of the resource file that contains the icon image for the page widget. For example, the value might be `images/myWidget_icon.png`. The image can be a .png file or a .gif file. This image represents the page widget in the Case Manager Builder palette.
Widgets/ runtimeClassName	Required	String	The class name for the page widget as specified in the runtime plug-in for the widget package.
Widgets/ previewThumbnail	Required	String	The relative path and name of the resource file that contains the thumbnail image for the page widget. For example, the value might be `images/myWidget_thnl.png`. The image can be a .png file or a .gif file. This image is not used in IBM Case Manager V5.2.

Related reference:
"Defining a page widget definition file" on page 156

"Defining an action definition file" on page 158
"Defining a property for a page widget or an action" on page 160
"Defining a property type" on page 161
"Defining a widget event" on page 163

Defining a page widget definition file

The page widget definition file is a JSON file that provides detailed information about a custom page widget. You must provide a definition file for each page widget in your custom widget package.

The following table describes the properties that are supported for a page widget definition file.

Table 95. Supported properties for page widget definition files

Property	Required or Optional	Type	Description
id	Required	String	A unique identifier for the page widget.
category	Required	String	The identifier of the category in which the page widget is to be listed in Case Manager Builder.
title	Required	String	The name to be displayed for the page widget in Case Manager Builder.
description	Required	String	A description of the page widget.
definition	Required	String	The full path and name of the definition file for the page widget.
preview	Required	String	The relative path and name of the resource file that contains the preview image for the page widget. For example, the value might be `images/myWidget_prv.png`. The image can be a .png file or a .gif file. This image is not used in IBM Case Manager V5.2.

Table 95. Supported properties for page widget definition files (continued)

Property	Required or Optional	Type	Description
icon	Required	String	The relative path and name of the resource file that contains the icon image for the page widget. For example, the value might be `images/myWidget_icon.png`. The image can be a .png file or a .gif file. This image represents the page widget in the Case Manager Builder palette.
runtimeClassName	Required	String	The class name for the page widget as specified in the runtime plug-in for the widget package.
previewThumbnail	Required	String	The relative path and name of the resource file that contains the thumbnail image for the page widget. For example, the value might be `images/myWidget_thnl.png`. The image can be a .png file or a .gif file. This image is not used in IBM Case Manager V5.2.
properties	Required	Array	An array that defines the properties that can be set for the page widget in Case Manager Builder.
events	Required	Array	An array that identifies the events that the page widget publishes and subscribes to.

The following example shows the structure of a page widget definition file. For examples of page widget properties, see “Defining a property for a page widget or an action” on page 160. For examples of page widget events, see “Defining a widget event” on page 163.

```
{
    "id":"CustomInbasket",
    "title":"Custom Inbasket",
    "category":"EducationWidgets",
    "description":"EN description of Custom Inbasket",
    "definition":"CustomInbasket.json",
    "preview":"images/custom/custominbasket_preview.gif",
    "icon":"images/custom/custominbasket_icon.gif",
    "runtimeClassName":"icm.pgwidget.inbasket.CustomInbasket",
    "previewThumbnail":"images/custom/custominbasket_thumb.gif",
    "properties":[
    ],
    "events":[

    ]
}
```

Related reference:

"Defining the widget package catalog file" on page 152

"Defining an action definition file"

"Defining a property for a page widget or an action" on page 160

"Defining a property type" on page 161

"Defining a widget event" on page 163

Defining an action definition file

The action definition file is a JSON file that provides detailed information about a custom action that is used for page widgets. You must provide a definition file for each custom action in your custom widget package.

When you develop an action, create a Java class that inherits from the `com.ibm.ecm.extension.PluginAction` class. In your class, override the `getAdditionalConfiguration()` method to return a JSON object.

The following example shows a JSON object:

```
{"ICM_ACTION_COMPATIBLE": true,
    "context": null,
    "name": "Custom Add Case Action",
    "description": "An action to add cases from other solution",
    "properties": [
       {
          "id": "label",
          "title": "Add a custom Case",
          "defaultValue": "Custom Add Case",
          "type": "string",
          "isLocalized":false
       },
       {
          "id": "solution",
          "title": "Solution",
          "type": "string",
          "isLocalized":false
       },
       {
          "id": "caseType",
          "title": "Case Type",
          "defaultValue": "",
          "type": "string",
          "isLocalized":false
       }
    ],
    "events":[
       {
            "id":"icm.OpenAddCasePage",
            "title":"Open Add custom Case Page",
            "direction":"published",
            "type":"broadcast",
            "description":"Open Add Custom Case Page"
        }
    ]
};
```

The following table describes the properties that are supported for an action definition file:

Table 96. Supported properties for an action definition file

Property	Required or Optional	Type	Description
ICM_ACTION _COMPATIBLE	Required	Boolean	Set to true if the action can be used in the IBM Case Manager action framework. This framework extends the IBM Content Navigator action framework to provide case-related functions. Always set this property to true for IBM Case Manager.
type	Optional	String	Indicates special processing for the action. The following values are valid for the type property: **`iterator`** Specify this value if the action is defined by using a method such as `getIterator()`. The method returns a series of items that are rendered as buttons or menu items. **`checkbox`** The action is rendered as a check box in the toolbar or menu. If you specify this value, you must also set the fieldname property value.
fieldname	Required	String	If the type property is set to checkbox, set this property to the identifier of a property that is defined in the properties array. If the type property is not set to checkbox, omit the fieldname property.
description	Required	String	A brief description of the action.
context	Required	Array	Indicates the contexts in which the action can be used. The array elements can take the following formats: **[["Context 1", "Context 2"]]** The action requires both Context 1 and Context 2 to run. **["Context 1", "Context 2"]** The action requires either Context 1 or Context 2 to run. **[[“Context 1”, “Context 2”],[“Context 1”, “Context 3”], “Context 4”]** The action requires Context 1 and Context 2 or Context 1 and Context 3 or Context 4 to run. **[]** The action does not require a context to run.
name	Required	String	The name that is displayed in the user interface for the action.
properties	Required	Array	The properties that a user can configure for an action in a toolbar or menu for a page widget or that are used internally by the action at run time.

Table 96. Supported properties for an action definition file (continued)

Property	Required or Optional	Type	Description
events	Required	Array	The events that are published by the action. This array can be empty if the action does not publish any events.

Related reference:

"Defining the widget package catalog file" on page 152

"Defining a page widget definition file" on page 156

"Defining a property for a page widget or an action"

"Defining a property type" on page 161

"Defining a widget event" on page 163

Defining a property for a page widget or an action

You can define a property for a custom page widget or a custom action in the definition file. The property is used to configure the page widget or action in Case Manager Builder.

The following table describes the properties that are supported for page widgets and actions:

Table 97. Supported properties for page widgets and actions

Property	Required or Optional	Type	Description
propertyType	Required	String	A value that indicates whether the property is a single property or a property group. **`property`** Specify this value if the property object contains a single property. **`group`** Specify this value if the property object contains a group of properties.
type	Required	String	A value that indicates the type of the property. For more information, see Defining a property type.
id	Required	String	A unique identifier for the property.
defaultValue	Optional	Depends on the property type	The default value for the property.
required	Required	Boolean	A value that is set to true if the property is required. By default, the property is set to false.

Table 97. Supported properties for page widgets and actions (continued)

Property	Required or Optional	Type	Description
visibility	Optional	Boolean	A value that is set to true if the property is visible. By default, the property is set to true.
title	Required	String	The label that is displayed for the property in Case Manager Builder.
remapNeeded	Optional	Boolean	A value that is set to true if the property value needs to be updated when the solution is imported and deployed to a production environment. This property applies only to a string property.
propertiesMember	Optional	Array	A definition of the properties within a property of type group.

Related reference:

"Defining the widget package catalog file" on page 152

"Defining a page widget definition file" on page 156

"Defining an action definition file" on page 158

"Defining a property type"

"Defining a widget event" on page 163

Defining a property type

You can set the type property for a page widget property or an action property.

Set the type property for a page widget property or an action property to one of the following values.

Table 98. Group property types

Property type	Description
Tab	Defines a new tab in the Edit Settings window.
Section	Defines a section that can be expanded and collapsed.
Dropdown	Defines a drop-down list that is used to select a value from a group of properties.
propertyPanel	Defines a content pane in which a group of properties are displayed.

Table 99. Generic property types

Property type	Description
Boolean	Defines a property that has a Boolean value.
Datetime	Defines a property that has a datetime value.
Float	Defines a property that has a float value.
Integer	Defines a property that has an integer value.
String	Defines a property that has a string value.

The following example of the properties is defined in the properties section of the page widget definition file as follows:

```
{
   "propertyType":"property",
   "type":"integer",
   "id":"integer1",
   "defaultValue":20,
   "required":false,
   "visibility":true,
   "title":"Integer property 1"
},
{
   "propertyType":"property",
   "type":"float",
   "id":"float1",
   "defaultValue":12.34,
   "required":false,
   "visibility":true,
   "title":"Float property 1"
},
{
   "propertyType":"property",
   "type":"boolean",
   "id":"boolean1",
   "defaultValue":false,
   "required":false,
   "visibility":true,
   "title":"Boolean property 1"
},
{
   "propertyType":"property",
   "type":"string",
   "id":"string1",
   "defaultValue":"default string",
   "required":false,
   "visibility":true,
   "title":"String property 1"
},
```

The following example shows a datetime property for a custom page widget. This property is defined in the properties section of the page widget definition file as follows:

```
{
   "propertyType":"property",
   "type":"datetime",
   "id":"datetime1",
   "defaultValue":"2013-05-01T03:00:00Z",
   "required":false,
   "visibility":true,
   "title":"Date Time 1"
},
```

Table 100. IBM Case Manager property types

Property type	Description
caseType	Displays an editor that enables users to select a case type in Case Manager Builder.
Choicelist	Displays a choice list for the property. This value can be used with other property types such as String.
contextualMenu	Displays an editor that enables users to edit a menu for a page widget in Case Manager Builder.
Label	Provides a read-only label that is displayed for the property. This value must be set to `label` for an action.
Order	Displays an editor that enables users to configure the order of the tabs in the Case Information widget.
Role	Displays a list of roles that are available in the solution from which the user can select.
Task	Displays a list of tasks that are available in the solution from which the user can select.
Textarea	Displays an input field in which the user can enter a text string.
Toolbar	Displays an editor that enables users to edit a toolbar for a page widget in Case Manager Builder.
View	Displays a selection list that contains all the views that are available for the case types in a solution.
viewList	Displays a list of case type-view pairs that enables the user to select a view. This property enables the user to add multiple views into the list. The output is a list of view identifiers.

Related reference:

"Defining the widget package catalog file" on page 152

"Defining a page widget definition file" on page 156

"Defining an action definition file" on page 158

"Defining a property for a page widget or an action" on page 160

"Defining a widget event"

Defining a widget event

You can define events as part of the page widget definition. You can define incoming events that provide handlers for events that are received by the page widget. You can also define outgoing events that are published by the page widget. Outgoing events can be either broadcast or wired.

The following table lists the properties that you define for an event.

Table 101. Event properties

Property	Required or Optional	Type	Description
id	Required	String	The unique identifier for the event.
title	Required	String	The title of the event or event handler.
functionName	Required	String	For an incoming event, the name of the function that handles the event. This property is not used for outgoing events.
direction	Required	String	Indicates whether the event is incoming or outgoing. Set to **subscribed** for an incoming event and **published** for an outgoing event.
type	Required	String	For an outgoing event, indicates whether the event is broadcast or wired. Set to **broadcast** for an event that is broadcast and set to **wiring** for an event that must be wired.
description	Optional	String	A description of the event. This text is used as hover help for the event in the Wiring window.

The following code shows how events are defined in the page widget definition file.

```
"events":[
                {
                    "id":"icm.RoleChanged",
                    "title":"Role selected",
                    "functionName":"handleReceiveRole",
                    "direction":"subscribed",
                    "description":"Update the In-baskets widget to display the
                    in-baskets that are associated with the specified role."
                },
                {
                    "id":"icm.SelectRow",
                    "title":"Row selected",
                    "functionName":"handleSelectRow",
                    "direction":"published",
                    "type":"wiring",
                    "description":"The user clicked a row or pressed enter
                    in the in-basket to select the work item."
                },
                {
                    "id":"icm.OpenCase",
                    "title":"Open Case",
                    "functionName":"handleOpenCase",
                    "direction":"published",
                    "type":"broadcast",
                    "description":"Open a case object."
                }
]
```

Related reference:
"Defining the widget package catalog file" on page 152
"Defining a page widget definition file" on page 156
"Defining an action definition file" on page 158
"Defining a property for a page widget or an action" on page 160
"Defining a property type" on page 161

Tips for sizing IBM Case Manager widgets

Some size settings for IBM Case Manager widgets can cause unexpected behavior at run time.

When you configure the widgets for your Case Manager Client application, be aware of the following tips:

- Set the height of the **Case List** widget to 100% or to a specific pixel value for **Cases** in Solution Pages. If you leave the height as an automatic setting, users are not able to access overflow search results.
- When you define the height of the **Case Information** widget in pixels, be sure to specify a smaller value than the height of the region. Otherwise, users encounter difficulty when they try to scroll.
- You might encounter unexpected behavior when you change a *Choice* property to a radio button set. If you change the **Group alignment** setting to **Horizontal**, the width of the property can retain the width of the original choice property. This causes some of the radio buttons to shift to a second line instead of using the width of the container. To resolve this issue, change the **Field** width setting to accommodate the width of your radio button set.
- You might encounter unexpected behavior when you try to change the width of a property. After you enter a value for the **Width** setting, you should be able to save the change. However, the **Save** button is not always enabled after you enter a value. To resolve this issue, use the Tab key to tab out of the field. This action enables the **Save** button, and it enables you to save the change.

Widget toolbar

Adding an event action to a widget toolbar or menu

You can add an event action to a toolbar or menu to trigger a custom event that is to be handled by a page widget.

For example, you might add an event action to the In-basket widget toolbar for a custom event that filters work items based on a predefined property value. You might add an event action to the Case Information widget document menu for a custom event that enables users to select and add a case document as an attachment to the case.

To add an event action to a widget toolbar or menu:

1. In Case Manager Builder, open the page that contains the widget in Page Designer.
2. Click the **Edit Settings** icon for the widget that you want to add the event action to.
3. Click the **Menus** or **Toolbars** tab and, if necessary, select the specific menu or toolbar to add the event action to.
4. Click the **Add Menu Item** icon or the **Add Button** icon.
5. From the Action list, select **Event Action**.
6. If you are adding an event action to a toolbar, select a position from the **Alignment** list.
7. For **Label**, enter the display name for the event action.
8. For **Menu Identifier**, enter an identifier that can be used by the event handler to determine the menu or toolbar that the event action is triggered from.
9. For **Event Name**, enter the name of the handler for this event.
10. From the **Event Type** list, select how to publish the event.

    Broadcast

    Select **Broadcast** if the event is received by any event that has a corresponding incoming event.

    Wiring

    Select **Wiring** if the event must be wired to an incoming event.
11. For **Show this event action**, enter a script that is run to determine whether the button or menu item for this event action is visible. If you do not enter a script, the button or menu item is always visible.
12. For **Enable this event action**, enter a script that is run to determine whether the button or menu item for this event action is enabled. If you do not enter a script, the button or menu item is always enabled.
13. Click **OK**.
14. Save and redeploy your solution.

Event action payload definition

The payload for an event action contains the following properties:

Table 102. Event action payload properties

Property	Description
menuId	Identifier that can be used by the event handler to identify the source of the event.
eventName	Name of the handler for the event.
eventType	Value that indicates how the event is published. This property is set to **`Broadcast`** if the event is received by any event that has a corresponding incoming event. This property is set to **`Wiring`** if the event must be wired to an incoming event.
actionContext	Action contexts that are set on the page widget that this event action is triggered from.

Example payloads

```
payload = {
menuId:     "customSearchMenu",
eventName:  "customSearchEvent",
eventType:  "broadcast",
Solution:  icm.model.Solution
}
```

In the following payload, the Folder and Document properties are arrays of `ecm.model.ContentItem` objects.

```
payload = {
menuId:        "customAttachMenu",
eventName:     "customAttachEvent",
eventType:     "broadcast",
Case:          icm.model.CaseEditable,
CurrentFolder: ecm.model.ContentItem,
ResultSet:     ecm.model.ResultSet,
Folder:        ecm.model.ContentItem+,
Document:      ecm.model.ContentItem+
}
```

Related tasks:

"Adding a script action to a widget toolbar or menu"

Adding a script action to a widget toolbar or menu

You can add a script action to run a custom script from a widget toolbar or menu. For example, you can add a script action to the Case Information widget toolbar that enables users to add the selected case documents as attachments to a case.

About this task

To add a script action to a widget toolbar or menu:

Procedure

1. In Case Manager Builder, open the page that contains the widget in Page Designer.

2. Click the **Edit Settings** icon for the widget that you want to add the script action to.
3. Click the **Menus** or **Toolbars** tab and, if necessary, select the specific menu or toolbar that you want to add the script action to.
4. Click the **Add Menu Item** icon or the **Add Button** icon.
5. From the **Action** list, select **Script Action**.
6. If you are adding a script action to a toolbar, select a position from the **Alignment** list.
7. For **Label**, enter the display name for the script action.
8. For **Execute**, enter the script to run when this script action is selected from the toolbar or menu.
9. Optional: For **Show this script action**, enter a script that is run to determine whether the button or menu item for this script action is visible. If you do not enter a script, the button or menu item is always visible.
10. Optional: For **Enable this script action**, enter a script that is run to determine whether the button or menu item for this script action is enabled. If you do not enter a script, the button or menu item is always enabled.
11. Click **OK**.
12. Save and redeploy your solution.

Example

The following example script action is intended to run in the context of the action implementation. For more information, see Class `icm.action.Action`.

```
var selectedDocuments = this.getActionContext("Document");
if (dojo.isArray(selectedDocuments))
{
   var i;
   for (i=0; i<selectedDocuments.length; i++)
   {
      // attach selected document: selectedDocuments[i]
      ...
   }
}
else
{
    // attach selected document: selectedDocuments
    ...
}
```

Related reference:

"Adding an event action to a widget toolbar or menu" on page 169

Notices

This information was developed for products and services that are offered in the USA.

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, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (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 websites are provided for convenience only and do not in any manner serve as an endorsement of those

Trademarks

Terms and conditions for product documentation

Applicability

Personal use

Commercial use

IBM Online Privacy Statement

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 cookies that collect each user’s user name for purposes of session management, authentication, and enhanced user usability. These cookies cannot be disabled.

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.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at www.ibm.com/privacy and IBM’s Online Privacy Statement at www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at www.ibm.com/software/info/product-privacy.

Index

Product Number: 5725-A15

SC19-3682-09