

IBM Case Manager  
Version 5.3.3

*User's Guide*





IBM Case Manager  
Version 5.3.3

*User's Guide*



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.

© **Copyright IBM Corporation 2010, 2018.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

## **Adding and deploying a case management solution . . . . . 1**

Adding a solution . . . . .	3
Adding and reusing properties . . . . .	4
Deleting case properties before deploying a solution . . . . .	6
Adding and selecting roles . . . . .	7
Roles in solutions. . . . .	8
Solution pages for roles. . . . .	8
Configuring in-baskets for roles . . . . .	9
Adding custom in-baskets to roles. . . . .	10
Adding and modifying document classes . . . . .	11
Document classes . . . . .	13
Adding and modifying business objects . . . . .	14
Business objects . . . . .	16
Adding and modifying case types . . . . .	17
Case types. . . . .	19
Creating and configuring stages for a case type . . . . .	20
Designing views. . . . .	21
Defining properties views . . . . .	22
Containers. . . . .	24
Adding workflow data fields and workgroups to the properties view . . . . .	27
Adding external properties to the properties view . . . . .	27
Property editor settings . . . . .	28
Properties . . . . .	29
Business rules . . . . .	33
Setting completion menu options . . . . .	36
Adding tasks . . . . .	38
Tasks in solutions . . . . .	40
Workflow tasks . . . . .	40
Custom workflow tasks . . . . .	42
To-do tasks . . . . .	42
Quick tasks . . . . .	43
Task preconditions . . . . .	43
Task initiation . . . . .	44
Task states. . . . .	45
Building new FileNet P8 processes to complete tasks . . . . .	45
Adding a workgroup to a task . . . . .	48
Adding attachments to a task . . . . .	50
Adding a swimlane to a task . . . . .	50
Adding a step to a task . . . . .	51
Creating routes in a workflow . . . . .	60
Adding workflows that are not associated with tasks . . . . .	66
Integrating business processes with IBM Case Manager solutions . . . . .	67
Adding an existing FileNet P8 process as a task . . . . .	68
Adding an existing process as a task . . . . .	70
Adding a task that runs an IBM Business Process Manager Advanced business process . . . . .	71

Designing views for to-do tasks . . . . .	73
Multiple user editing of solutions . . . . .	74
Designing the case management client application . . . . .	75
Creating a custom page . . . . .	76
Pages . . . . .	78
Solution pages . . . . .	78
Case Details pages . . . . .	80
Add Case pages . . . . .	82
Split Case pages . . . . .	84
Add Task pages . . . . .	85
Work Details pages . . . . .	87
Custom Task Details pages . . . . .	90
Widgets . . . . .	91
Attachments widget . . . . .	93
Calendar widget. . . . .	96
Case Information widget . . . . .	97
Case List widget . . . . .	101
Case Stages widget . . . . .	103
Case Toolbar widget . . . . .	103
Chart widget . . . . .	105
Content List widget . . . . .	105
Form widget . . . . .	108
In-baskets widget . . . . .	112
Instruction widget . . . . .	114
Markup widget. . . . .	115
Original Case Properties widget . . . . .	116
Process History widget . . . . .	116
Properties widget . . . . .	117
Script Adapter widget . . . . .	120
Search widget . . . . .	122
Select Case Documents widget . . . . .	123
Split Case Properties widget . . . . .	123
Timeline Visualizer widget . . . . .	123
To-Do List widget . . . . .	124
Toolbar widget . . . . .	124
Viewer widget . . . . .	125
Website Viewer widget . . . . .	126
Work Item Toolbar widget . . . . .	126
Widget, action, and page events and wiring . . . . .	128
Action events . . . . .	128
Attachments widget events. . . . .	140
Attachments widget outgoing events . . . . .	140
Attachments widget incoming events . . . . .	141
Calendar widget events . . . . .	142
Calendar widget incoming events . . . . .	142
Case Information widget events . . . . .	143
Case Information widget outgoing events . . . . .	143
Case Information widget incoming events . . . . .	145
Case List widget events . . . . .	147
Case List widget outgoing events. . . . .	147
Case List widget incoming events . . . . .	148
Case Stages widget events . . . . .	150
Case Stages widget incoming events. . . . .	150
Case Toolbar widget events. . . . .	151
Case Toolbar widget incoming events . . . . .	151
Chart widget events . . . . .	152

Chart widget incoming events . . . . .	153
Content List widget events . . . . .	153
Content List widget outgoing events . . . . .	153
Content List widget incoming events . . . . .	154
Form widget events . . . . .	156
Form widget outgoing events . . . . .	156
Form widget incoming events . . . . .	157
In-baskets widget events . . . . .	159
In-baskets widget outgoing events . . . . .	159
In-baskets widget incoming events . . . . .	161
Instruction widget events . . . . .	163
Instruction widget incoming events . . . . .	163
Markup widget events . . . . .	164
Markup widget incoming events . . . . .	164
Original Case Properties widget events . . . . .	164
Original Case Properties widget incoming events . . . . .	164
Page Container widget events . . . . .	165
Page Container widget outgoing events . . . . .	165
Page Container widget incoming events . . . . .	169
Process History widget events . . . . .	172
Process History widget incoming events . . . . .	172
Properties widget events . . . . .	173
Properties widget outgoing events . . . . .	173
Properties widget incoming events . . . . .	182
Script Adapter widget events . . . . .	184
Script Adapter widget outgoing events . . . . .	185
Script Adapter widget incoming events . . . . .	185
Search widget events . . . . .	185
Search widget outgoing events . . . . .	185
Search widget incoming events . . . . .	186
Select Case Documents widget events . . . . .	186
Select Case Documents widget incoming events . . . . .	187
Split Case Properties widget events . . . . .	187
Split Case Properties widget outgoing events . . . . .	188
Split Case Properties widget incoming events . . . . .	195
Timeline Visualizer widget events . . . . .	197
Timeline Visualizer widget outgoing events . . . . .	197
Timeline Visualizer widget incoming events . . . . .	198
To-Do List widget events . . . . .	199
To-Do List widget incoming events . . . . .	200
Toolbar widget events . . . . .	202
Viewer widget events . . . . .	202
Viewer widget incoming events . . . . .	202
Website Viewer widget events . . . . .	203
Website Viewer widget incoming events . . . . .	203
Work Item Toolbar widget events . . . . .	204
Work Item Toolbar widget incoming events . . . . .	204
Testing your solution . . . . .	205

<b>Enhancing solution designs by using Process Designer . . . . .</b>	<b>207</b>
Primary queue . . . . .	207

Creating more than one in-basket in Process Designer . . . . .	208
Assigning a primary queue to more than one role . . . . .	209
Disassociating a role's primary queue . . . . .	210
Defining a custom role . . . . .	210
Changing primary queue definitions after upgrade . . . . .	211
Adding case operations to a task . . . . .	212
addCaseOwners operation . . . . .	214
addCaseRoleMembers operation . . . . .	214
addCommentToCurrentCase operation . . . . .	214
addCommentToCurrentDocument operation . . . . .	215
addCommentToCurrentTask operation . . . . .	215
addDependentObject operation . . . . .	216
completeCurrentCaseStage operation . . . . .	216
createCasePackage operation . . . . .	216
createCaseUsingSameCaseType operation . . . . .	218
createCaseUsingSpecifiedCaseType operation . . . . .	219
createDefaultCasePackage operation . . . . .	220
createDiscretionaryTaskInCurrentCase operation . . . . .	220
createDiscretionaryTaskInCurrentCaseWithWorkflowParams operation . . . . .	220
createDiscretionaryTaskWithProps operation . . . . .	221
createNewDependentObject operation . . . . .	223
createSubfolderUnderCurrentCase operation . . . . .	223
fileAttachmentsToCurrentCase operation . . . . .	223
findDependentObjects operation . . . . .	224
getCasePropertyNames operation . . . . .	225
getCasePropertyValues operation . . . . .	225
getCaseStructure operation . . . . .	226
getTaskPropertyNames operation . . . . .	227
getTaskPropertyValues operation . . . . .	227
placeCurrentCaseStageOnHold operation . . . . .	227
relateCurrentCase operation . . . . .	228
releaseCurrentOnHoldCaseStage operation . . . . .	228
removeCaseOwners operation . . . . .	229
removeCaseRoleMembers operation . . . . .	229
removeDependentObject operation . . . . .	229
replaceCaseOwners operation . . . . .	229
replaceCaseRoleMembers operation . . . . .	230
restartPreviousCaseStage operation . . . . .	230
searchTasks operation . . . . .	231
setCasePropertyValues operation . . . . .	231
setDependentObjectProperties operation . . . . .	231
setTaskPropertyValues operation . . . . .	232
terminateTasksInCurrentCase operation . . . . .	232
unfileAttachmentFromCurrentCase operation . . . . .	233
unfileDocAttachmentFromCurrentCase operation . . . . .	233
unrelateCurrentCase operation . . . . .	234

<b>Notices . . . . .</b>	<b>235</b>
Trademarks . . . . .	237
Terms and conditions for product documentation . . . . .	237
IBM Online Privacy Statement . . . . .	238

<b>Index . . . . .</b>	<b>239</b>
------------------------	------------

---

## Adding and deploying a case management solution

You can use the Case Manager Builder to add a solution that case workers will access from the Case Manager Client. A *solution* is a set of web pages, content, and process definitions that provide a framework so that you can manage cases.

### About this task

A solution can include roles, document classes, properties, in-baskets, and case types. A case type contains business rules, views, and tasks that must be completed to close the case.

The wizard guides you through creating a solution and adding case types, roles, and document classes. After these solution artifacts are added, you can easily add tasks and steps to complete the tasks in the case type section of the user interface.

Each task can contain a workflow that must be followed to complete the task. A workflow consists of steps connected by a route, which is defined by connectors. An existing workflow can be reused from a business process management system.

Steps are assigned to roles or workgroups, or are system instantiated, or can be further defined in Process Designer. A step is defined by properties, attachments, workgroups, or data fields.

As you define the case type, you can set document classes as preconditions in tasks and assign steps to roles. In addition, you can extend property details for the case type. For example, when you create a case type for credit card disputes, you can add the existing property template for account ID so that the case type includes the account number. In addition, you can set the dispute form as a precondition for an opening a dispute claim task.

You can also define the name for the personal in-basket and how to display the properties for the work items in the in-basket. You can configure a role to have one of the following types of in-baskets:

- A role based in-basket, which is optional.
- A personal in-basket, which can be common, or based on a role.
- An **All assigned work** in-basket.

A role in-basket differs from the personal in-basket in that it contains work items that are assigned to a role, but not to a specific person. Work items in a role in-basket can be accepted by any user to work on. The **All assigned work** in-basket displays a list of all the open work items that are assigned to users, and only users in specific roles can access this in-basket.

You can create business rules to implement business policies and practices, such as determine process routing or update case properties if particular conditions are met. After you create rules, you can use them in a workflow by adding rule steps.

After you create the case types and steps for the solution, decide how you want views for the cases and work items to be shown in Case Manager Client. For the views, you determine which properties are shown and the grouping for the properties.

You can assign case properties to the case summary, case data, and search views that case workers access in the Case Manager Client. For example, you might want case workers to be able to search on the case ID or due date properties.

You can customize the layout of the pages that display in the Case Manager Client application.

Solutions can be edited by more than one user at the same time. Save and close, then commit your changes to make the solution assets available to other users and for deployment.

If Case Manager Builder is integrated with a version control system (VCS), committing your changes copies the changes to the VCS. You then deliver the solution to create in the VCS a baseline or snapshot of the solution with the changes applied.

When you finish designing the solution, you can preview the solution by deploying it, and then test the solution in the development environment. You can make adjustments in Case Manager Builder and deploy the solution in the development environment as much as you need to.

After you are satisfied that the solution is correct, ask the IT administrator to deploy the solution to a production environment.

**Important:** Each user who might add a solution must use their own user ID. If you edit the same solution in more than one browser session, changes might be lost.

## Procedure

To build and deploy a solution:

1. Add a solution.
2. Add roles, document classes, and properties to the solution.
3. Add case types to the solution.
4. Add properties, views, case folders, business rules, and tasks to the case types.
5. Add steps to the tasks.
6. Design the client application by customizing the page layouts.
7. Save and close, then commit your changes for the solution. If Case Manager Builder is integrated with a VCS, you are prompted to add comments and, possibly, to provide parameter values to be used to check the changes into the VCS.
8. If Case Manager Builder is integrated with a VCS, deliver the changes for the solution.
9. Deploy the solution to your development environment and test it in the Case Manager Client. You must assign a user to your roles to fully test your solution.
10. Request that your IT administrator deploy the solution into a production environment with the IBM® Case Manager administration client. Your IT administrator will configure security for the case types in IBM FileNet® Enterprise Manager.



## Results

After the solution is deployed, users can log in to the solution in Case Manager Client.


“Adding a solution”

“Multiple user editing of solutions” on page 74


“Designing the case management client application” on page 75

“Testing your solution” on page 205

### Related tasks:

 [Upgrading solutions](#)

### Related information:

 [Designing your case management solution](#)

---

## Adding a solution

A solution maps to an instance of the Case Manager Client that users log in to so that they can manage their cases. A solution consists of one or more related case types that provide the documents, data, business processing, and routing to the case workers. For example, a solution for a human resources department might include a case type for new hires, a case type for retirement, and a case type for resource action.

### About this task

You can add a solution without using a template, add a solution from a template that was provided by an administrator, or copy an existing solution. you can also use the wizard to create a solution.

If a template includes existing assets, such as properties or document classes, the prefixes are retained from the template. The new prefix will apply to new assets that are created for the solution. In addition, the template provider might choose to prevent you from changing or deleting properties, document classes, or case types. Check with the template provider for specific restrictions.

Each user who might add a solution must use his or her own user ID. If you edit the same solution in more than one browser session, changes might be lost.

### Procedure

To add a solution:

1. On the Manage Solutions page, select the method that you want to use to create a solution:
  - Select a template from the menu and click **Add Solution**. Select **No template** to create a solution without using a template.
  - Select **Use the wizard to define the solution** if you want guidance in creating the solution, and then click **Add Solution**.
  - Select an existing solution and click **Copy**.

**Tip:** If you do not find an existing solution that you wanted to copy on the Manage Solutions page, the solution might be assigned to a different project area. Check with your system administrator.

2. Enter a name and solution prefix. The solution prefix is a 2 - 5 character prefix that is prepended to all case types, document classes, properties, and task unique identifier names that are created for the solution. The solution prefix must be unique across all solutions. However, if you create a solution from a template, reused properties that are part of the template retain the prefix that is assigned by the template supplier.

**Restriction:** You cannot modify the solution name or prefix after you create the solution.

3. Optional: Add a solution description and icon. If you copied a solution, you can change the icon after you create the solution. You can modify the solution description by clicking the **Edit solution description** icon in the solution summary on the solution home page.

Solution icons are a predefined set of icons.

4. Create case types, roles, and document classes and set the in-basket views to complete the solution. If you selected the **Use the wizard to define the solution**, the wizard displays a checklist that walks you through these steps. You can also add properties and pages to the solution.

“Adding and reusing properties”

“Adding and selecting roles” on page 7

“Adding and modifying document classes” on page 11

“Adding and modifying business objects” on page 14

“Adding and modifying case types” on page 17

“Creating and configuring stages for a case type” on page 20

“Designing views” on page 21

“Business rules” on page 33

“Adding tasks” on page 38

## Adding and reusing properties

You define properties, such as names or dates, at the solution level and reuse them in any case type, document class, task, or step in that solution.

### About this task

You can create new properties using Case Manager Builder, or reuse properties that already exist in the target object store.

You cannot modify reused properties, and they are not recreated during solution deployment. You can select from a list of reusable properties, which can be sorted or filtered by name, unique ID, or type.

### Procedure

To add or reuse properties:

- Add to the solution:
  1. From the solution home page, open the Properties page. Then, click **Add Property** and select either **Reuse Property** or **New**.

Reused properties are properties that have been defined in a solution that has been deployed. Reused properties can be used again in other solutions. For example, if you are building a solution for automobile insurance, you might need properties for a policy number and for a customer name in most case documents in the solution. You can create the policy number and

customer name properties as templates to quickly add the properties where you need them throughout the solution.

**Important:** You can reuse String, Integer, Boolean, Float, and datetime type properties. You cannot reuse a property that is of type Business Object. You cannot use multi-value properties that are configured as unique and unordered or as hierarchical choice lists.

\

To reduce the amount of space that is used by properties, try to reuse existing properties, if possible. When you select **Reuse Property**, you can select one or more properties that already exist in Content Platform Engine.

If necessary, create new properties. To reduce the amount of space that is used by properties, see the tips in *Minimize database row sizes*.

2. Define property values. The property value fields that are displayed depend on the type of the property. For example, single properties can have a default property value. Some properties, like float and integer properties, can have minimum and maximum values.
3. Optional: If you define a property as a choice list, you can use the choice list from a reused property that you added to the solution if the type matches. To create a new choice list, on the Properties page click **Manage Choice Lists** and define choice list values.

**Important:** IBM Case Manager does not prevent you from entering duplicate values in a choice list. For example, assume you add *First* as a choice item and assign the value as 1. You then add *Primary* as another choice item and assign the value as 1. In Case Manager Client, both *First* and *Primary* appear in the choice list.

**Restriction:** If the solution was created from a template that included a choice list, the choice list is read-only. You cannot change the choice list. You can create a new choice list with the same or different values.

If a choice list is updated in the solution where it was originally created, and the solution is redeployed, the change is propagated to the solutions that reuse the choice list. When the solution that reuses the choice list is opened in Case Manager Builder, the solution designer is prompted to update the choice list to the version that has been redeployed to the target object store.

- Add to a case, document class, or task:
  1. When you create a case type, document class, or task, click **Add Property**. You can select an existing property that is defined for the solution, reuse a property, or create one by clicking **New**.
  2. Define property values. The property value fields that are displayed depend on the type of the property. For example, single properties can have a default property value. Some properties, like float and integer properties, can have minimum and maximum values. If you are editing a property that existed in the solution, any values that you enter here apply only to this case type or document class. For example, you can override the default value that was set at the solution level.

Properties that you define at the case type, document class, or task level apply only to the case type, document class, or task that you are currently designing. For example, when you create a case type for automobile claims, you can add the existing reused property for policy number so that the case type includes the policy number.

Only case properties are available in the case views that case workers see in the client application.

You can assign a case property as the **Case Title Property** on the Views page. The selected property must be a string or integer data type and a single value property. The selected property must not be hidden. If you do not assign a case property, the Case Title Property is the Case ID by default.

- Add one or more task preconditions:
  1. When you create or edit a task, click the **Preconditions** tab.
  2. Select a precondition type. Depending on the task, you can select:
    - A case property is updated**  
To configure this precondition, select a property in the **Case properties** list.
    - A document is filed in the case**  
To configure this precondition, select **Any document class** or select one or more types in the **Document classes** list. You can make this task repeatable.
    - A property condition is met**  
To complete this precondition, click **Add Condition** to select a case property that can be included in an expression. For example, you can set a precondition for the claim value property so that if the claim value exceeds a specific value, then the fraud assessment task starts.
  3. To add another precondition in which a property condition must be met, click **Add Condition**. You can add other conditions, such as the AND or OR conditions.

## What to do next


When designing your solution, if you reuse a long string property from Content Engine, do not expose that long string property in the Search View of a case type. If you expose long string properties in your case type Search View, case workers will receive an error when they search using that long string property. The **begins with**, **ends with**, **contains**, **is empty**, and **is not empty** operators are the only valid operators for a long string property, and if a case worker uses a different operator in the client, the search fails.

“Deleting case properties before deploying a solution”

### Related tasks:

“Assigning properties to a step” on page 56

### Related information:

 [Creating a property template](#)

## Deleting case properties before deploying a solution

If you delete properties from a solution before deploying the solution, you can also remove the properties from the Process Engine solution configuration file before deploying the solution. You cannot delete the properties after the solution is deployed.

## About this task

You might want to delete properties that are not used in a solution. If you do not remove the deleted properties from the solution configuration file in Process Engine and the Process Configuration Console, and then deploy the solution, the deleted properties become data fields in the solution and in the task map in

Process Designer. The extraneous data fields might confuse case workers and leave unnecessary columns in an in-basket.

## Procedure

To delete case properties:

1. Delete the properties in Case Manager Builder by clicking the **Remove** icon for each property.
2. From the Manage Solutions page, open your solution in Process Designer by clicking **More Actions > Open Process Designer**.
3. Open the configuration console by clicking **View > Configuration**. Do not open the configuration console by clicking **Tools**.
4. Expand the **Work Queues** node and edit the Role queue properties.
5. Remove properties that were added to an In-basket:
  - a. Click the **Data Fields** tab and delete the properties that you deleted from the solution in Case Manager Builder.
  - b. Expand the **User Queues** node and edit the inbox to delete the properties that you deleted from the solution in Case Manager Builder.
6. On the **Tasks** tab, click the **Assignments** tab, select **After Completion**, and delete any field assignments for the deleted properties.
7. Click the **Workflow Properties > Data Fields**, and remove the deleted properties.
8. Validate, save, and close the solution.

### Related information:

Cannot access Process Designer from Case Manager Builder with Google Chrome

## Adding and selecting roles

A role defines and groups case workers by the type of work that they do. You can associate roles with steps in the tasks of a solution.

### About this task

You can add roles to a new solution by using the solution wizard or by using the solution home page. If you add roles to a new solution by using the solution wizard, each role is assigned a default personal in-basket and a role in-basket. To add or associate other in-baskets with a role, complete the wizard and open the solution.

## Procedure

To add a role from the solution home page:

1. In the Roles page of the solution home page, click **Add Role**.
2. Specify a role name and optionally a description. The role name must be 55 characters or less and it cannot start with the tilde (~) special character. The role name cannot use the same name as a task in the same solution.
3. Select the type of personal in-basket and the user features to display for the role in Case Manager Client.
4. On the **Pages** tab, assign the solution page for the role, then click **OK**.
5. Configure the in-baskets for the role. For more information, see “Configuring in-baskets for roles” on page 9 and “Adding custom in-baskets to roles” on page 10.

6. Associate the role with task steps by adding a step and assigning it to a role in the Step Designer.

## What to do next

After you add roles, you can deploy the solution to the development environment and test the roles by adding users or groups.

“Roles in solutions”

“Solution pages for roles”

“Configuring in-baskets for roles” on page 9

“Adding custom in-baskets to roles” on page 10

### Related tasks:

“Designing views” on page 21

## Roles in solutions

You define roles at the solution level. You then associate roles with steps in tasks. You assign users and groups to roles in the Case Manager Client to specify which users can access a particular task or step.

Roles can be shared by all the cases in the solution. For example, you define roles, such as Clerk, Approver, and so on, to organize the groups of users who must process different types of tasks in the application.

**Restriction:** You can define only one in-basket per role in the Case Manager Builder. You can also define a personal in-basket and get access to the inbasket that contains all of the assigned work items. An in-basket filters work items in a queue to display only items that answer a specific query. If you want to define additional in-baskets, you must use the FileNet Process Designer.

Because you define roles at the solution level, you can reuse those roles across all case types when you design the solution. You can also choose to create a role that is applied only to a specific case type. When you create tasks or steps, you can assign roles to the task or step from roles that are already defined for the solution, or you can create a new role for the solution.

After you create the solution, you assign users and groups defined in your LDAP server to roles in the Case Manager Client. For example, some roles that might create for an automobile claims solution are field agent, insurance agent, claims adjuster, supervisor, or fraud investigator. The roles define which in-baskets and Case Detail pages a case worker sees when using Case Manager Client.

### Related tasks:

“Configuring in-baskets for roles” on page 9

### Related information:

Cannot access Process Designer from Case Manager Builder with Google Chrome

## Solution pages for roles

Solution pages provide case workers with access to cases and work items that are in a solution. By default, the two IBM Case Manager Solution pages, Cases and Work, are used for every role.

The Solution pages are displayed when a user first opens the solution in Case Manager Client. These pages remain open during the user's session.

You can create custom Solution pages to meet the needs of a specific role. You then associate the custom pages with that role.

**Related concepts:**

“Solution pages” on page 78

**Related tasks:**

“Creating a custom page” on page 76

## **Configuring in-baskets for roles**

An in-basket lists the steps for the cases in a solution. (In Case Manager Client, the steps that are defined for a task are called work items.) In Case Manager Builder you can configure one or more in-baskets for a role. .

### **About this task**

IBM Case Manager provides three types of in-baskets: role, personal, and assignment. You can also define other types of in-baskets by using IBM Process Designer

#### **Role in-baskets**

Case Manager Builder automatically create a role in-basket when you create a role. In Case Manager Client, this in-basket appears on a tab that is named after the role and contains all the steps that are assigned to a specific role for all the open cases in the solution.

By default, the following system properties are included in any role in-basket: Subject, Time Created, Step Name.

#### **Personal in-basket**

Optionally, you can configure your solution to display a personal in-basket for a case worker. You can configure the in-basket to display the properties that are common to all roles that are defined for the solution. Alternatively, you can configure the in-basket to display properties that are defined for the specific role to which a case worker belongs. Work items are added from a role in-basket to a personal in-basket when one of these actions occurs:

- A case worker moves a work item from the common role in-basket to a personal in-basket.
- A case worker reassigns a work item to a different case worker.
- An automated processing step assigns the item to a specific workgroup.

In Case Manager Client, this in-basket appears on the My Work tab and contains the steps that are assigned to that case worker.

By default, the following system properties are included in any new personal in-basket: Subject, Time Created, Step Name.

#### **Assignment in-basket**

An assignment in-basket shows all of the work items that are assigned to all case workers in every role. For example, you can assign this in-basket to a manager or supervisor role in addition to a personal in-baskets. To view and reassign work from the Assignment in-basket, a user must have write access to the application space for the solution. You configure this access after you deploy the solution by using the Process Configuration Console.

By default, the following system properties are included in any Assignment in-basket: Subject, Time Created, Step Name, Assigned to.

## Procedure

To configure in-baskets for a role:

1. Click the **Roles** tab and then click the role for which you want to assign an in-basket.
2. Select the type of personal in-basket you want displayed for the role:

In-basket type	Description
<b>Personal (Common): Show the common view</b>	Displays the step properties that are common to all roles in the solution.
<b>Personal (Role): Show a custom view for this role</b>	Displays the step properties that are defined for the specific role to which the case worker belongs.

If you do not want to display a personal in-basket, select **Do not show common or role personal in-baskets**.


3. Optional: Select the work assignment check boxes to enable role members to move work items into the personal in-basket or to reassign work items to other case workers.
4. Optional: To display the assignment in-basket for this role, select the **Show the in-basket that displays all assigned work** check box.
5. Configure the properties and filters to be displayed in the in-baskets:
  - a. Click the **In-baskets** tab and then click the in-basket that you want to configure.
  - b. Click the **In-baskets General** tab and select the properties to be displayed. You can specify the order that the properties are listed, whether each property is sortable, and you can limit the properties that are displayed so that only the most important information for the role is displayed in the in-basket. The case worker must open the work item to view the additional properties. For example, an automobile accident claim case has 25 properties. For quick access, you select these five properties to display in the in-basket: claim number, client family name, client given name, loss date, and estimated loss value.
  - c. Click the **In-basket Filters** tab. Define the filters that your users can use to filter steps (work items in Case Manager Client). Filtering reduces what users see in their in-basket for this role. For example, you might want to create a filter that is named High Priority for users to quickly see high priority work items.

If you define a filter with the operator **is like**, the filter returns all items that include the filter string anywhere in the value.
  - d. Click **OK**.

### Related concepts:

“Roles in solutions” on page 8

### Related tasks:

 Customizing the case unique identifier prefix

## Adding custom in-baskets to roles

You can add new in-baskets to existing queues from Process Designer into your IBM Case Manager solution. For example, you might want to provide different views of a queue, such as loan applications for over or under certain amounts, for different purposes.



## About this task

By default, Case Manager Builder creates one *Role In-basket* for each role and, optionally, one personal in-basket. You can define additional custom in-baskets, but you must add them to the solution and assign them to a role by using Process Designer as described in this task.


**Important:** You must assign the new in-baskets to a role for the new in-baskets to be displayed in Case Manager Client.

## Procedure

To add additional in-baskets to roles:

1. Open the solution in Process Designer by clicking **File > Solution > Edit** and select the solution definition file.
2. Select a case type and click **View > In-baskets**.
3. Select a role or inbox queue from the **Queue for in-baskets** menu.
4. Add a new in-basket by clicking the **Add** icon.
5. Specify a name, select the fields, and optionally make fields sortable. To include system fields, select **Show system fields** on the **Add fields to in-basket** page.
6. Optional: On the Create Filters page, add a filter.
7. Assign the new in-basket to a role by clicking **View > Roles**, selecting a role, and clicking the **Add** icon. You can also select other in-baskets.
8. Validate, save, and close the solution.

### Related tasks:

 [Define in-baskets](#)

## Adding and modifying document classes

Document classes organize and classify the documents that belong to a case. For example, you can create a document class for case correspondence and another document class for photographs and images. You assign document classes at the solution level. You can assign a document class to start a new case or as a precondition for a task.

## About this task

You can add document classes to a new solution by using the wizard or from the solution home page. If you add document classes from the solution home page, you can create a new document class based on an existing document class. The new document class inherits all of the properties that are defined for the parent document class.

**Tip:** Document classes that are based on an existing document class are nested under the parent document class in the list of document classes for the solution. Expand a parent document class to view the child document classes.

You can also reuse document classes that are defined for other solutions. Reused document classes can only be edited in the original source solution.

## Procedure

To add or modify a document class from the solution home page:

1. Click the **Document Classes** tab.
2. Select an existing document class to modify or create a document class.

Option	Description
<b>To add a document class</b>	<ol style="list-style-type: none"> <li>1. If you want the new document class to inherit properties from an existing document class, select the parent document class from the list of document classes. If you do not want the new document class to inherit properties from an existing document class, ensure that no document class is selected in the list.</li> <li>2. Click <b>Add Document Class &gt; New</b>.</li> <li>3. Enter a name and a description for the document class. The <b>Unique Identifier</b> field is updated as you enter the name for the document class. You cannot change the unique identifier after you click <b>OK</b>.</li> </ol>
<b>To add a document subclass</b>	<ol style="list-style-type: none"> <li>1. Select the parent document class in the list of document classes.</li> <li>2. Click <b>Add Document Class &gt; New subclass</b>.</li> <li>3. Enter a name and a description for the document class. The <b>Unique Identifier</b> field is updated as you enter the name for the document class. You cannot change the unique identifier after you click <b>OK</b>.</li> <li>4. Click <b>OK</b>.</li> <li>5. On the Properties tab, add the properties that apply to this document subclass.</li> </ol> <p><b>Tip:</b> To view the properties that a document class inherited from a parent document class, select the <b>View inherited properties</b> check box on the <b>Properties</b> tab for the child document class.</p>
<b>To modify an existing document class</b>	Click the document class name in the list.
<b>To reuse a document class</b>	<ol style="list-style-type: none"> <li>1. Click <b>Add Document Class &gt; Reuse Document Class</b>.</li> <li>2. Select a document class that exists in Content Engine.</li> </ol> <p><b>Restriction:</b> You cannot modify a reused document class. Properties of a reused document class are not displayed in Case Manager Builder, and you cannot add new properties to the reused document class.</p>

3. Click **OK**.
4. Optional: Add properties to the document class.

- Click **Save**. Case Manager Builder does not save your edits automatically as you work. When you click **OK** to dismiss a dialog box or an editor view, your selections are stored in memory. You must click **Save** to store your changes to the solution definition file. You can click **Save and Close** to save and close the solution or **Close**. If you do not save your changes, you can save or discard the changes before you close the solution.

“Document classes”

**Related tasks:**

“Adding and reusing properties” on page 4

**Document classes**

Document classes help you to organize and classify the documents that belong to a case. You can provide additional information about the documents by assigning properties to the document class.

You define document classes to group similar documents and the information about the documents that are related to the case. You can create as many document classes and properties as needed.

For example, an automobile claim case might have the document classes and properties shown in Table 1.

*Table 1. Examples of automobile claim document classes and properties*

Document class and description	Properties related to this document class
Policy documents, such as the written automobile policy and formal changes to the policy	Deductible amount, effective date of the policy, expiration date of policy, policy number, policy type, vehicle identification
Application forms, such as the initial written application form	Customer address, customer given name, customer family name, customer phone number, date of application, policy number, policy type, vehicle identification
Claim forms, such as a claim for a broken windshield	Claim number, customer given name, customer family name, date of loss, policy number, type of loss, vehicle identification
Repair estimation documents, such as an estimate for a new windshield from a glass repair company	Claim number, customer given name, customer family name, estimate total, repair item, vendor name,
Damage assessment documents, such as the report from insurance claim evaluator or photographs of the damaged windshield	Claim number, customer given name, customer family name, date of loss, vehicle identification, vehicle replacement value
Correspondence, such as letters sent to the customer about the claim	Claim number, customer given name, customer family name, date of loss, vehicle identification
Proof of ownership, such as the automobile registration from a state licensing agency or a bill of sale	Claim number, customer given name, customer family name, date of loss, vehicle identification, vehicle license plate number

When you create a document class, you assign a name and description for the document class. The name is displayed to case workers in the Case Manager Client. Case Manager Builder automatically creates a unique identifier for the document class.

You can assign one or more properties that provide information about the document to a document class, and you can assign the same property to more than one document class. In the automobile claim example, several document classes use the claim number property to associate a document with a specific claim number.

If properties are required for multiple document classes, create the properties at the solution level so that you can quickly add them to each document class. Although a property can be shared by many document classes within a solution, you can define unique values for the default value, required, and hidden settings for each document class. The values that you set at the document class level override any settings at the solution level.

If you want to set the Read-only value for a document class, you must do so at the Step Properties level of the task by using Step Designer.

For example, for the policy document class in the example, the policy number property is a required property because each policy must have a unique number. For the application form document class, the policy number is not known when the application form is submitted, and the policy number is added later after the application is processed. The policy number is not a required property for the application form document class.

You can also reuse document classes that were imported into the development environment from the IBM FileNet P8 domain where you will be deploying this solution into production. You cannot modify reused document classes and they are not re-created during solution deployment.

You can create a document class that is based on an existing document class. The new child document class inherits all the properties that are defined for the parent document class. For the document classes listed in Table 1, you might first create a parent claim document with the following properties: Claim number, customer given name, customer family name. You could then create the document classes for claim forms, repair estimations, correspondence, and proof of ownership based on the parent document. Note that you cannot customize the inherited properties for a child document class.

**Related tasks:**

“Adding and reusing properties” on page 4

## **Adding and modifying business objects**

A case can contain multiple instances of a solution entity that is best represented by a collection of properties. In this scenario, you can define a business object that is a structured data type that contains a collection of properties. You define a case property in your solution that uses this business object as the data type.

### **About this task**

**Important:** Business objects are supported only in Content Platform Engine, V5.5.0 or later, repositories. Business objects are not supported in earlier versions of Content Platform Engine repositories or in IBM Content Manager repositories.

You can create a business object as a child of an existing business object. The child inherits all the properties that are defined for the parent business object.

You can also reuse business objects that are defined for other solutions. Reused business objects can be edited only in the original source solution.

## Procedure

To add or modify a business object from the solution home page:

1. Click the **Business Objects** tab.
2. Create a business object or modify an existing object.

Option	Description
To add a business object	<ol style="list-style-type: none"> <li>1. Click <b>Add Business Object &gt; New</b>.</li> <li>2. Enter a name and a description for the business object. The <b>Unique Identifier</b> field is updated as you enter the name for the business object. You cannot change the unique identifier after you click <b>OK</b>.</li> <li>3. If you do not want this business object to be used as a property type, select the <b>Do not use this business object as a property type</b> check box.</li> <li>4. Click <b>OK</b>.</li> </ol>
To add a business object as a subclass of an existing business object	<ol style="list-style-type: none"> <li>1. Select the parent business object from the list of business objects.</li> <li>2. Click <b>Add Business Object &gt; New subclass</b>.</li> <li>3. Enter a name and a description for the business object. The <b>Unique Identifier</b> field is updated as you enter the name for the business object. You cannot change the unique identifier after you click <b>OK</b>.</li> <li>4. If you do not want this business object to be used as a property type, select the <b>Do not use this business object as a property type</b> check box.</li> <li>5. Click <b>OK</b>.</li> </ol> <p><b>Tip:</b> To view the properties that a business object inherits from a parent business object, select the <b>Show inherited properties</b> check box on the <b>Properties</b> tab for the child business object.</p>
To modify an existing business object	Click the business object name in the list.

3. On the **Properties** tab, add or modify the properties that define this business object.
4. Optional: Select a property from the **title** list to uniquely identify instances of this business object.

A business object is a subclass of the Content Platform Engine `DependentObject` class. An object that is instantiated from a subclass of the `DependentObject` class is a dependent object, which is non-addressable. A dependent object does not possess a unique identifier by which it can be referenced and; therefore, can be identified only by value. The **title** field identifies this value.

**Tip:** IBM Case Manager does not check at run time to ensure that the business object title is unique. However, you can write a script to validate this information. For information about scripts to validate data, see *ICM Data Validation When Adding New Cases*.

If a business object is not referenced at run time, you do not need to set a title. You might not set a title for a parent business object that is never used as a

property type. However, if you do set a title, the children inherit that title. You can override that inherited title by selecting a different property for a child business object.

## What to do next

Create case properties with the type set to Business Object and select this business object from the **Business Object** list. In most situations, you can use this business object property as you would use any other case property. For example, you can define a precondition to start a task when a business object property is modified. However, you cannot use a business object in any of the following situations:

- In an expression for the **A property condition is met** option in a task precondition
- In an in-basket
- In Step Designer as a property for a step in a task workflow
- In a business rule
- As a document property

Like other case properties, a business object property is added to the default view for any case type in which the property is used. A business object property is presented in a table in which each column represents a property that is contained in the business object.

“Business objects”

## Business objects

At times, you need a collection of properties to represent a single solution entity. If a case contains multiple instances of such a solution entity, you can use a business object, which is a structured data type that represents a solution entity as a collection of properties. You can then assign that business object as the data type of a multivalued property. For example, to represent a person who is covered by an insurance policy, you need properties for given name, surname, ID, birth date, relation to policy holder, and similar information. You create a business object, Insured Party, with these properties.

A business object is a subclass of the Content Platform Engine `DependentObject` class. Objects that are instantiated from a subclass of the `DependentObject` class, such as a business object, are dependent objects that do not have unique identifiers. You can access a dependent object only as a property of an independent object. These facts result in the following requirements for a business object:

- To be able to reference a specific instance of the business object, you select one of the business object properties as a title. In the example business object, Insured Party, you select the ID property as the title because it is the one property that uniquely identifies the person.
- To use the business object in a solution, you must create a multivalued case property that is assigned the business object as its data type. For the example business object, Insured Party, you then can create a case property, Insured Parties, and assign this property a type of Insured Party. A case worker then can enter information for each person in the Insured Parties table.

If you have business objects that have several properties in common, you can create a parent business object that contains all the common properties. You then create subclasses to contain the unique properties for each business object. For example, you might have business objects for different types of insurance policies.

Some properties apply to all policy types, such as policy ID, customer ID, and agent. However, each policy type has unique properties. In this situation, you can create a parent business object, Policy Object, that is assigned all the common properties. You then define subclasses for Policy Object and assign these subclasses the unique properties. The subclasses inherit all the properties that are assigned to the parent.

The following table provides an example of a parent and child structure:

Business object	Parent	Properties
Policy Object	None	Policy ID Customer ID Agent
Vehicle Policy Object	Policy Object	Policy ID (inherited from parent) Customer ID (inherited from parent) Agent (inherited from parent) Vehicle Type Vehicle Make Vehicle Model Vehicle ID
Property Policy Object	Policy Object	Policy ID (inherited from parent) Customer ID (inherited from parent) Agent (inherited from parent) Property Type Street Address City State

In this example, you can set the title to Policy ID on the Policy Object parent. The children inherit the title unless you choose to override it by selecting a different property.

Because the Policy Object object does not contain complete information for a policy, you do not want to use this object as a case property type. To prevent users from selecting the Policy Object object as a case property type, select the **Do not use this business object as a property type** check box for this object. Then, the child business objects show in the list, but the Policy Object object does not.

## Adding and modifying case types

A solution can have one or more case types. You can add case types to a new solution by using the wizard or from a solution home page. You can modify existing case types from the solution home page.

### About this task

Case types identify the tasks, content, processes, and views that are required to manage the case. For example, a solution for a human resources department might include a case type for new hires, a case type for retirement, and a case type for resource actions. Each case type can have one or more tasks that must be completed to process and close the case, such as reviewing a claim application or distributing a check for an approved claim. Tasks include steps that are displayed

as work items in case worker in-baskets. You can create business rules to implement business policies and practices, such as determine process routing or update case properties if particular conditions are met. You design views to control what information case workers see when they work on a case.

**Tip:** To edit the solution in Process Designer, find the solution on the Manage Solutions page and click **More > Open Process Designer**. From Process Designer, you can edit workflows, roles, in-baskets, and configuration objects for your case types.

## Procedure

To add or modify a case type from the solution home page:

1. Click the **Case Types** tab and add a case type, or select an existing case type to modify.
2. Enter a name and description for the case type. If you are adding a case type, the **Case type unique identifier** value is updated as you enter the name to create a unique identifier for the case type. You cannot change this value. However, you can change how it is displayed to case workers in Case Manager Client by using IBM FileNet Enterprise Manager after the solution is deployed to production.
3. Optional: Select the class of document that triggers a new case. For example, when a new claim form document is added to the system, a new case starts to process the claim.  
To copy matching document properties to case type properties, select **Map document class properties**. For example, if a new claim form document includes the policy number, that policy number can be copied into the policy number property for the new case.
4. Optional: If you want users to be able to create quick tasks for a case, select **Enable case workers to create quick tasks**.
5. Optional: If you want users to be able to create custom tasks for a case, select **Enable case workers to create custom tasks**.
6. Optional: If you want users to be able to add documents or attachments from external repositories, select **Allow documents and attachments from repositories other than the case management object stores**. If you plan to include content from an external repository, configure a connection to the repository in IBM Content Navigator before you deploy the solution.
7. Optional: Select the Case Manager Client pages that you want to display when a user adds a case, splits a case, or views a case. When you create a solution, the only available pages are the default pages. After you deploy the solution to the development environment, you can use Case Manager Client to add pages for the solution and return here to select a different page.
8. Optional: If you want a different Case Details page to display in Case Manager Client for a specific role, add that role and select the page. When you create a solution, the only available page is **Default Case Details page**. After you deploy the solution to the development environment, you can use Case Manager Client to add pages for the solution and return here to select a different page.
9. Add properties, views, case folders, business rules, and tasks to the case type.
10. Click **Save**. Case Manager Builder does not save your edits automatically as you work. When you click **OK** to dismiss a dialog box or an editor view, your selections are stored in memory. You must click **Save** to store your changes to the solution definition file. You can click **Save and Close** to save and close the



solution or **Close**. If you do not save your changes, you can save or discard the changes before you close the solution.

11. Click **Validate**. The case type design is validated and a message is displayed in the status bar.
12. On the Manage Solutions page, for your solution, click **Commit**. Committing your changes makes the solution assets available for further editing or deployment.


“Case types”


**Related concepts:**

“Business rules” on page 33

**Related tasks:**

“Adding and reusing properties” on page 4

 [Planning for security](#)

 [Connecting and configuring additional repositories](#)

## Case types

*Case types* define the tasks, the necessary document classes to support the task, the task steps, and the roles that must complete those steps to solve a business problem. The case type also includes properties that are displayed to case workers in views. Related case types make up a solution.

An example of case type is a loan application. A loan application case in the Case Manager Client contains detailed information such as correspondence, tasks, policies, and events that case workers or case teams collaborate on to resolve and close that case. Case workers and case managers work together on cases.

For each case type you define the following data:

### Case Type information

You can define attributes such as the name and description for each case type. The system assigns a unique identifier for the case type.

You can specify a starting document class. When a document of this class is added to the repository, a new instance of this case is automatically created.

You can enable custom task creation, which allows end users to design additional tasks for the case.

You can enable users to add documents and attachments that are in a repository other than the case management object stores.

For the Add Case, Split Case, and Case Details pages, you can choose which of the page types to use as the default layout for that page.

### Properties

You can assign properties to the case type, and you can decide which properties are displayed in the client views. You can assign existing properties that were defined for the solution, or you can add new properties.

**Views** The case views define the properties and their display order in the Case Manager Client. A single property can be used in one or more of the case views.

The summary view is shown in the Case Information widget, and the properties that you specify are displayed in the search results.

The view for the Search widget controls which properties the user can search against.

For the Properties widget, you can create properties views that define how case properties are laid out in the Case Client interface.

If you do not define views, IBM Case Manager uses a default layout.

### Case folders

You can assign an empty folder structure to which the case workers can add documents that are required to complete the case. Folders provide logical groupings for documents that are related to the case. You can create more than one top-level folder, and you can create subfolders for any folder.

When cases are designed with a predefined subfolder structure that includes a large number of folders, the case can take a longer time to create and initialize. To maintain a reasonable response time during case creation, limit the complexity of your predefined folder structure for a case to ten or fewer subfolders.

If your solution requires cases with a more complex subfolder structure, the best practice is to create the subfolders later, after the case is initialized. You can add the subfolders programmatically, within an automatic task process, or on demand by a user as needed.

Folder names can contain 255 characters. The folder name cannot contain the following characters: {\ \* / < > : | ? "}

**Restriction:** Character limits might vary depending on your language.

**Rules** You can create business rules to implement business policies and practices, such as determine process routing or update case properties if particular conditions are met. After you create rules, you can use them in a workflow by adding rule steps.

**Tasks** Each case type contains one or more tasks that can include a workflow process to complete that task. A task has one or more steps that must be completed in order to complete the task. A case is not complete until all required tasks are completed or manually disabled.

#### Related concepts:

“Tasks in solutions” on page 40


“Business rules” on page 33

#### Related tasks:

“Adding tasks” on page 38

“Adding and reusing properties” on page 4

“Designing views” on page 21

 [Planning for security](#)

## Creating and configuring stages for a case type

You define stages for a case type to represent the lifecycle of a case. The first stage starts automatically when the case is started. When one stage completes, the next stage automatically begins. For example, for a loan case type, you might have an initial review stage that starts when a client submits a loan application. When the initial review verifies that all the necessary information is submitted, the case moves on to the credit check stage.

## Procedure

To create and configure case stages:

1. After you create a case type, open the Stages page and click **Add Stage**.
2. Enter a name, a unique identifier, and, optionally, a duration for the stage.
3. Repeat steps 1 and 2 for each case stage that you add to the case type. The sequence in which the case stages occur is determined by the order in which the stages are listed on the Stages page. That is, the first stage in the list is the first stage to start. When that stage completes, the second stage in the list starts.  
To change the sequence in which a case stage occurs, click the **Move Up** or **Move Down** arrow to move that stage. Alternatively, you can drag the stage in the new position.
4. To display case stage information in Case Manager Client, add the Case Stages widget to the Case Details page. For more information about this widget, see “Case Stages widget” on page 103.
5. Configure the ways in which state of the case stage is changed at run time. You can change the case stage state in the following ways:
  - Add the following actions to a menu or toolbar in Case List widget or Case Toolbar widget:
    - Complete Stage
    - Restart Stage
    - Toggle Stage
  - Add stage steps to System lane of the workflow for a task.
  - Implement the following case operations:
    - `completeCurrentCaseStage`
    - `placeCurrentCaseStageOnHold`
    - `releaseCurrentOnHoldCaseStage`
    - `restartPreviousCaseStage`

### Related tasks:

“Adding a step to a task” on page 51

### Related reference:

“`completeCurrentCaseStage` operation” on page 216

“`placeCurrentCaseStageOnHold` operation” on page 227

“`releaseCurrentOnHoldCaseStage` operation” on page 228

“`restartPreviousCaseStage` operation” on page 230

## Designing views

You design views to specify the properties that are displayed to case workers for a case type. For each case type, you can specify the properties that are displayed for the case summary and the properties that are searchable. In addition, you can specify the properties that are displayed in the Properties widgets that case workers use to complete cases and work items.

### Before you begin

On the Properties page for the case type, add the properties that are used for the case type.

## About this task

You can specify the following views for a case type:

### Case Summary

Use the Case Summary view to select and order the properties that are displayed in the **Summary** tab of the Case Information widget and in the Case List widget.

In addition to the case properties, you can select predefined properties such as the case identifier, case type, and date modified to include in this view.

### Properties Layout

Use the Properties Layout view to define different layouts for the Properties widgets that are used for the case type. For example, you can define different layouts to meet the requirements of different roles or tasks. Each layout can display a different set of properties.

You can use the system-generated view for the Properties widget that is provided by IBM Case Manager. Alternatively, you can specify a custom view as the default for the case type. The default view is used unless you configure the Properties widget on a page to use a specific view. For example, you might configure the Properties widget on a Case Details page to use a view for a specific role. You might configure the Properties widgets on an Add Task page to use a different layout for a specific step.

### Case Search

Use the Case Search view to select and order the properties that are available for building a search in the Search widget.

In addition to the case properties, you can select predefined properties such as the case identifier, case type, and date modified to include in this view.

You can also specify a case property to use as the title of cases of this case type. On the Views page, click the **Edit** icon for the **Case Title Property** field. Select a single-value string or integer property to use as the case title. If you do not specify a property for the case title, the Case ID is used by default.

*“Defining properties views”*

### Related concepts:

*“Summary view for the Case Information widget” on page 101*

*“Case List widget” on page 101*

*“Properties widget” on page 117*

*“Search widget” on page 122*

## Defining properties views

You can define views that provide different layouts for the Properties widget in a case type. For example, you can define different layouts to meet the requirements of different roles or tasks. You can then specify which view of the Properties widget is to be used on a specific page.

## About this task

Defining a view for the Properties widget is optional. You can use a system-generated view instead. The system-generated view provides an ordered list of all properties that are associated with the case type.

As you design a view, consider the space that is available in the user interface. If the Properties widget is too wide or too high, you might see scroll bars that make editing the properties awkward for the user. Use the different layout containers such as the tabbed layout container or the titled layout container to organize the content of the view efficiently.

**Restriction:** If you set a custom size for the Properties View Designer window, minimizing or maximizing the window causes View Designer to display incorrectly.

## Procedure

To define a view for the Properties widget:

1. On the Views page for the case type, click **Add View** on the **Properties Layout** tab.
2. Enter a name and an identifier for the layout and then click **OK**.
3. To make this view the default layout for the case type, select the view from the **Default view** list. This view is then used for the Properties widget on all pages for this case type unless you select a different layout for the widget on a specific page. If you do not select a default view, a system-generated view is used as the default.

**Tip:** To reduce the amount of configuration, select the view that is used most frequently as the default view.

4. Click the **Open Properties View Designer** icon for the view.
5. Drag one or more containers onto the canvas. Configure the settings for the containers as needed. Each container provides a different layout for the properties that it contains. For the layout containers, you can add other containers and properties to a container.

**Note:** In the solution designer, not everything in the canvas works in run time as it does during design time. Use the canvas only as a tool for designing your layout, not to test runtime behavior.

While you drag a container, press the Ctrl key to copy the container instead of moving it. Press the ESC key to cancel a drag operation.

**Tip:** To make working with the containers easier, the design view displays extra space around the containers. Click the **Hide Extra Padding** icon to see an approximation of how the view will look in Case Manager Client.

6. Select the category of properties to display from the drop-down list in the **Properties** palette. Drag the properties into the appropriate containers. Use the Shift and Ctrl keys to select multiple properties. While you drag a property, press the Ctrl key to copy the property instead of moving it. Press the ESC key to cancel a drag operation.

You can also drag properties from one container to another on the palette.

The categories of properties include case properties, runtime properties, and, if defined for the solution, task properties. The runtime properties include workflow fields and external properties.

To add workflow data fields or workgroups to the view, click **Run time only > Workflow Field**. Alternatively for some containers, you can select **Data fields** or **Workgroups** from the **Automatically include** list to include all the workflow data fields or workgroups that are exposed in a step to the container.

7. With the property selected, configure the settings for that property.

You must configure a workflow field, workgroup, or external property so that Case Manager Client can discover the data field or workgroup at run time.

When you add a property to a container, it initially uses the default control for editing the property. For example, a Boolean property is displayed as a check box by default. You can change the type of control that is used for a property. For example, you might change the control so that a Boolean property displays as radio buttons or a drop-down list. You can also select **Static text** to display the property value in read-only mode and configure the appearance and behavior of some controls.

**Important:** You can set a default value for the property in the view. However, this value is applied only on the Add Case page, and only if the property value is null and is not specified by another mechanism such as a JavaScript call or an external data service.

8. Save the view and close the designer.

“Containers”

“Adding workflow data fields and workgroups to the properties view” on page 27

“Adding external properties to the properties view” on page 27

“Property editor settings” on page 28

“Properties” on page 29

### **Containers:**

You use the containers in Properties View Designer to organize the content of the view. The three layout containers provide for the general layout of the view and can contain both properties and other containers. The two property containers provide specific layouts for properties.

#### **Layout container**

This container provides a simple, rectangular container that can contain any type of property or container. You can configure the layout container to display the contents either horizontally or vertically. You can also configure the position of labels to display either beside or above the property editor.

#### **Multiple column layout container**

This container consists of a series of columns. You can click the **Insert** button and **Delete** button to add or remove columns. Each column contains a layout container that can contain any type of property or container and can be configured separately.

You can drag columns to reorder them in the container. You can also drag a layout container from elsewhere in the view or from the container palette into the multiple column layout container. Similarly, you can drag a column from the multiple column layout container to another location in the view.

### **Titled layout container**

This container consists of a layout container that has a title bar and that can contain any type of property or container. You can configure the titled layout container so that it can be expanded or collapsed at run time.

### **Property list container**

This container can contain any type of property except for a property whose type is business object. The container displays the properties in a vertical list with the labels beside the property editor.

### **Property table container**

This container consists of a table in which each column contains a multivalued property. Unlike other containers, the property table container groups the content in addition to organizing the layout.

**Restriction:** Single-value properties, properties of type business object, and workgroups are not supported in Property Table containers.

The property table container binds the property values in a row so that each row is treated as a logical data set. IBM Case Manager groups each data set and stores the content in the multivalued properties in the same order as they appear in the table rows. Case workers can change this order by using the up and down arrows in the table to reorder the rows. Because of this grouping, IBM Case Manager puts the following restrictions on a property table:

- Values in a property table cannot be empty or null. Therefore, all properties in a property table container, except for string properties, are required.
- Case workers can enter only a single value in a property table cell. They cannot enter multiple values for a property in the same row.

You can include multivalued properties that get values from an external data service in a property table. However, problems can occur if the values that are returned for one property depend on the value of another property in the same property table and the external data service returns the values as dynamic choice lists. In this situation, the choice list can contain values that do not apply to a specific instance of the dependent property in the table.

If one of the properties in a property table container is read-only, the toolbar is not available for the table at run time. A case worker cannot add a row, delete a row, or move a row. However, a case worker can double-click any property in an existing row that is not set to read-only and edit the value for that property.

“Container width and height”

*Container width and height:*

You configure the width and height of containers in Properties View Designer to best fit the layout of your page.

When you first open a view in Properties View Designer, the canvas contains a single layout container. This container is the root container in which you add other containers and properties.

Each container is as wide as its parent container if a width and height is not specified. You can set a custom width.

The default height is determined by the content of the container. You can override the default height by setting the height to a specific percentage or number of pixels. If you set the container to a specific height (percentage), you must also set a specific height for the parent container or containers, including the root container. If the height of the content exceeds the specified height, a vertical scroll bar is displayed.

#### **Tabbed Layout containers**

- The Tabbed Layout container always fills the full width of the container unless an explicit height and width are specified.
- All tab panes within the container have the same width regardless of their explicit width settings.
- The height can be set for the tab container or for the individual tab panes within the container.
- If the content of a tab pane exceeds the specified width and height of the tab container, horizontal or vertical scroll bars are displayed.

#### **Titled Layout containers**

- The Titled Layout container expands to the height of its content. When the container is collapsed, the height is reduced to the height of the title bar.
- If an explicit height is specified, the container height is always that height. When the container is collapsed, the content is hidden but the height is not reduced.

#### **Multiple column containers**

- The default width for each container is an equal percentage of the parent container.
- When you override any column settings, the default settings are discarded and you must specify the settings for all columns.
- If the total column width exceeds the container width, a horizontal scroll bar is displayed.

#### **Property List containers**

- Labels have the width equal to the longest label in the list.
- The column for the editor gets the remaining width.
- A field width percentage is the percentage of the editor column.
- The field width can be specified for each property. If a width is not specified, a default width is used.

#### **Property Table containers**

- You can specify the width of the table. If the combined column width exceeds the container width, a horizontal scroll bar is displayed.
- You can specify the height of the table. If the height is not specified, the table increases in height without a limit. If a height is specified, the table height is fixed and a vertical scroll bar is displayed in the table as necessary.
- You can specify the field width for each property. If a width is not specified, a default width is selected. Percentage-based field widths are specified as a percentage of the table width.



## Adding workflow data fields and workgroups to the properties view:

You can associate data fields and workgroups with the workflows that you define for tasks by using either Case Manager Builder or Process Designer. You can add these data fields and workgroups to a properties view in the same way that you add case properties.

### About this task

You can add data fields and workgroups to a view one at a time. By using this method, you can define settings such as labels and user interface controls for these fields.

You can also use the **Automatically include** setting to add to a container all the data fields or workgroups that are exposed in a step. By using this method, you can ensure that all data fields or all workgroups for the step are included in the view. However, you cannot define settings for specific fields. Instead, the default settings are used for each field.

To have a data field or workgroup displayed in the Properties widget at run time, you must associate that data field or workgroup with the task and step that a user is performing. If these conditions are not met, the field is not displayed in the view.

### Procedure

To add a single data field or workgroup to a container in Properties View Designer:

1. In the **Properties** palette, select **Run time only** from the drop-down list. Then, drag **Workflow Field** into the container.
2. Select the workflow field and edit the settings:
  - a. Enter the name of the data field or workgroup exactly as it is defined in the workflow.
  - b. Select the data type of the field. For a workgroup, select **Workgroup**.  
You can leave the data type as **Unspecified**. In this situation, Case Manager Client determines the data type at run time and uses the appropriate default control for editing the field.
  - c. If the data field or workgroup can have multiple values, select the **Multiple values** check box.
  - d. Specify the property settings for the workflow field. The property settings depend on the data type of the field.

**Tip:** At runtime, the workflow system assigns values to all of the workflow data fields. Use Process Designer if you want to specify a default value for workflow fields.

### Related concepts:

 [Workflow properties - data fields](#)

## Adding external properties to the properties view:

You can add properties to a properties view that are managed externally from your solution. For example, external properties can be from another object model or provided by an external data service.

## About this task

The processing of external properties is not managed by IBM Case Manager. Therefore, if your view includes external properties, you must ensure that the mechanism for processing these properties is in place. For example, you might use a script to manage the properties and include a Script Adapter widget on any page that uses this view for the Properties widget.

## Procedure

To add an external property to a container in Properties View Designer:

1. In the **Properties** palette, select **Run time only** from the drop-down list. Then, drag **External Property** into the container.
2. Select the external property and edit the settings:
  - a. Enter the collection identifier and property identifier that Case Manager Client will use to locate the property at run time.
  - b. Select the data type of the external property and define the property settings as needed.

You can leave the data type as **Unspecified**. In this situation, Case Manager Client determines the data type at run time and uses the appropriate default control for editing the property.

## Property editor settings:

When you add a property to a view in Properties View Designer, you can select the type of editor that Case Manager Client users use to enter or select property values. For example, you might specify that an integer property is edited by using either a number text box or a number spinner. You can also specify that the property value is displayed as static text, which prevents users from editing the value.

The editors that are available for a property depend on the following criteria:

- The data type of the property
- Whether the property is single-valued or multivalued
- Whether a choice list is associated with the property
- The type of container in which you place the property

In addition, the editor that you select might have related settings. For example, if you select **Date text box** for a DateTime property, the **Include time portion** check box is shown in the settings. If you select this check box, the time is shown in the same text box as the date. If you select **Select, Filtering select**, or **Radio button set** for a Boolean property, the **False when empty** check box is shown. If you select this check box, the property value is set to false if the user does not select a value.

By default, if you select **Filtering select**, the Filtering select editor includes an empty value in the choices. Clear the **Include empty choice** check box to remove the empty choice.

## Editors for multivalued properties

You define two editors for a multivalued property. The first editor determines the editing of the property as a whole. The second editor determines the editing of the individual values within the multivalued property.

For most containers, you set the editor for the multivalued property as a whole in the **Editor Settings** area. The following table describes two editors that are available for all multivalued properties. If a choice list is associated with the property, other editors are also available depending on the data type of the property.

Editor	Description
Inline editable list	This option provides for faster entry of property values. The user can press Enter to enter the next value. However, the user cannot edit the value in the list. Instead, the user must delete an incorrect value and then enter the correct value. Values are always added to end of the list. The user can move the value up and down.
Editable entry list	This option provides more flexibility in editing a list. The user can modify values and insert values anywhere in the list. However, it is a bit slower to add values because the user must click the <b>Add</b> button to add each value.

If you add a multivalued property to a **Property Table** container, the first editor is set automatically to be an inline editable list.

You set the editor for the individual values in a multivalued property in the **Value Editor Settings** area. The editors that are available depend on the data type of the property.

### Properties:

You can configure various property settings in Properties View Designer, such as the width of the property editors.

### Width for property editors

The default label width is 30%. The label width cannot be specified if the horizontal alignment is set for the container.

If you use the **Long format** or **Full format** setting for a DateTime property, you might need to adjust the width of the setting to accommodate the longest long or full format value of the supported locales.

IBM Case Manager enforces the following behavior:

- Adjusts the specified width to avoid unnecessary scroll bars
- Provides a default field width for each property editor if no width is specified
- Provides a minimum field width for each property editor if the specified width is less than the minimum field width

### Patterns for Integer and DateTime fields

The canvas in the Properties View Designer shows you how the selected editor setting looks in Case Manager Client.

To specify a pattern for Integer or DateTime fields, you must use a valid format as defined in Unicode Technical Standard #35:

- For Integer fields, IBM Case Manager supports the patterns that are listed in Part 3: Number Format Patterns.

- For DateTime fields, IBM Case Manager supports the patterns that are listed in Part 8: Date Format Patterns.

**Tip:** Check your property in the deployed solution to verify that the pattern that you used is valid.

### Displaying DateTime fields in coordinated universal time (UTC)

By default, DateTime values are displayed according to the user's local time. You can select the **Display in UTC** check box to display a DateTime value in coordinated universal time. If you select this check box, you must use either the short format or medium format for the time.

Other widgets do not display DateTime values in coordinated universal time. For example, assume that the same DateTime property appears in both the Properties widget and the Case Information widget on a page. If the Properties widget is configured to display the property in coordinated universal time, the property can appear to have different values.

### Reconciliation of conflicting property settings

The attributes of a property are determined by the view settings and by the settings in the controller layer. The view settings are specified in Properties View Designer. The controller settings are set either when you define the property in the solution or programmatically by an agent such as an external data system, a script, or web service.

IBM Case Manager reconciles any difference in the setting of a property attribute in a view definition and in the controller layer as follows:

- Any property attribute in the controller layer that is set by using an external data service or some other form of automation is automatically applied to the view. The corresponding property editor setting in the view definition is ignored.
- For property attributes that are not set by automation, IBM Case Manager applies one of the following rules to reconcile any differences in a setting:
  1. Precedence is given to the view setting for settings such as label, which affect only the display of the property.
  2. Precedence is given to the controller setting for settings such as pattern, which affect the property data.
  3. View settings can never be less restrictive than the editor settings. For example, if the controller maximum value is 100 and the view maximum value is 50, then the view setting is applied. But if the controller maximum value is 100 and the view maximum value is 150, the controller setting is applied.

The following tables identify the precedence rules that IBM Case Manager applies to each setting:

*Table 2. General property settings*

Setting	Precedence rule
Label	1
Required	3
Hidden	3

Table 2. General property settings (continued)

Setting	Precedence rule
Read-only	3

Table 3. DateTime property settings

Setting	Precedence rule
Format	1
Date pattern	1
Time pattern	1
Minimum value	3
Maximum value	3

Table 4. Number property settings

Setting	Precedence rule
Zero if empty	3
Round automatically	3
Pattern	2
Decimal places	3
Minimum value	3
Maximum value	3

Table 5. String property settings

Setting	Precedence rule
Maximum length	3
Minimum length	3
Truncate automatically	3
Capitalization	2
Adjust capitalization	3
Pattern	2

Table 6. Boolean property settings

Setting	Precedence rule
False if empty	3

### Best practices for property editor settings

If you set a property attribute dynamically by using an external data service or some other form of automation, do not configure the property editor setting in the view definition. The setting must be configured in the view definition to use the default value.

If you do not set a property attribute dynamically, use the view definition to customize the property editor settings.

Whether you set a property attribute dynamically or through customized settings in the view definition, do not specify a setting that exceeds the constraints of the object store. For example, if the object store imposes a maximum length of 10 for a string property, do not set the length to 20 in the view definition.

### Multivalue editors and property tables

Use the **Radio button set** as the editor setting for a multivalue property only if the property has a limited number of choices. When a multivalue property uses the **Radio button set** editor setting, the field appears as a drop-down list in Case Manager Client. When the user clicks the list, a dialog box opens. The user then clicks the **Add** icon to open the editor that displays the choices as radio buttons. If the number of choices exceeds the size of the dialog box, a scroll bar is displayed. However, the scroll bar does not work correctly.

If a multivalue property contains more than a few choices, consider using an editable entry list as the editor setting.

At run time, multivalue editors behave in the following ways:

- You must enter at least one value for the property, even if that value is an empty string. However, if you are using an Oracle database, you can not enter empty strings in a multivalue property.
- The required tooltip is always displayed. The user must click the Exit button (x) or press the ESC key to close it.
- Entry of invalid data is supported, but you cannot save the data until all of the invalid data is corrected.
- Double-click to change to edit mode. In Mozilla Firefox, the property field is automatically editable when you add a row. In Microsoft Internet Explorer, you must double-click to change to edit mode.
- When the property field is in edit mode, you must use the arrow keys to edit at a specific location.
- When you use the hint inside the editor, the hint flashes. It is recommended to use a hint position outside the editor.
- For multivalue editors only: Clicking any other area in the view does not close the multivalue editor window.

### Boolean property

If you want check box editor settings to behave like other editor settings, you can use the **Uniform labels** setting in the Layout and Titled Layout container objects.

- Set the **Layout direction** setting to **Horizontal** and the **Label position** to **Beside** for a Layout container.
- When you drag a Boolean property, the label appears on the right side whereas the label for the other types of properties appears on the left.
- Setting the label position to **Above** has no effect on the Boolean property.

### Filtering properties

You can use the filter in the property palette to filter the properties list.

## Cross-validating constraints

Properties View Designer validates the value type that you enter for a setting. However, Properties View Designer does not cross validate that value with the constraints that are set by other settings for the same property. For example, you can select the **Zero when empty** check box for a property, and then also set **Minimum value** to 1. You can save this combination in Properties View Designer. However, at run time, Case Manager Client automatically displays an error state for the field because it defaults to zero and the minimum value is 1.

## Editor behavior

Check boxes always force the null value to false.

At run time, editors behave in the following ways:

- Radio buttons cannot be cleared to set to null after a selection is made.
- DateTime: Delete the date and time value to set it to null. You can enter date and time value by using the keyboard without using the picker.
- DateTime: If the date is entered first, the time defaults to 12:00 PM. If the time is entered first, the date defaults to the current date.
- If a property is marked both required and read-only, the asterisk character (\*) is shown.
- The asterisk character is not shown if the **Zero when empty** check box or **False when empty** check box is selected. You can use these settings to suppress the required asterisk if zero or false is a suitable default value for the property. Because the property always has a value, the required asterisk is not required.
- Empty rows are not supported in any data type except strings. If you are using an Oracle database, empty rows are not supported for strings.

## Rendering of editor settings that are shared by multiple tasks

If a property is added to more than one task, and the property is configured the same way for each task, the widget is rendered as configured. However, if the property is configured differently for each task, the widget is rendered with the least restrictive setting as follows:

1. Required: false
2. Hidden: false
3. Default value: null

## Business rules

Business rules determine the actions to take if particular conditions are met. After you create business rules for your case type, you can use the business rules in a task to determine process routing or to update case properties.

You can include user-defined case properties and case system properties in your rules. If you want the rule to refer to data that is external to the case, such as the credit rating of a loan applicant, you can define custom rule parameters and then include them in the rule.

To use the business rule in your workflow, add a rule step to a task by using the Case Manager Builder Step Designer. To associate the rule step with the business rule, set the **Rule Name** property of the rule step. If your business rule uses

custom parameters, you must edit the rule step in Process Designer to map the custom rule parameters to the external data sources.

## Text-based business rules

A text-based business rule consists of the following elements:

### Definitions

The optional definitions part of the rule specifies variables that can be used in the rule. The scope of these variables is limited to the specific rule in which the variables are defined.

### Conditions

The if part of the rule specifies the circumstances in which the actions in the then and else parts of a rule are to be carried out.

**Restriction:** You can use the following system properties in business rule conditions, but you cannot include them in business rule actions to set their values:

- Case Identifier
- Creator
- Date Created
- Date Last Modified
- Last Modifier
- Case State

### Actions

The then part of the rule specifies the actions to take if the conditions are true. The optional else part of the rule specifies actions to take if the conditions in the if part are false.

**Tip:** For troubleshooting purposes, you can specify one or more print actions in a rule so that the associated rule step returns the specified strings to the workflow. The return value of the rule step can then be mapped to a data field that can be used by the next step in the workflow.

For example, for a loan case type, you can define a rule to set the value of the Interest\_Rate case property to 3.5% if the credit rating of the applicant is higher than 700. In the following examples, Credit\_Rating is a custom rule parameter. (The following examples are written in the English locale.)

```
if Credit_Rating is more than 700
then set the Interest_Rate of LoanCaseType to 3.5;
```

## Table-based business rules

A table-based rule, also known as a decision table, consists of condition and action columns. Each condition column specifies the circumstance for which particular actions are to be carried out. Each action column specifies the actions to take if the conditions are true. Each row in the table represents a separate rule. If the conditions of a row are met, the actions in that row are taken.

For example, you want to create a group of rules that sets the value of the Interest\_Rate case property for different ranges of credit rating values. You first define a condition column that represents the value of the Credit\_Rating custom rule parameter for a loan applicant.

```
Credit_Rating is <a number>
```



Next, you define an action column that represents the interest rate to offer:  
set the Interest\_Rate of LoanCaseType to <a number>

Then, in each row, you specify a particular credit rating and the corresponding interest rate:

Credit rating	Interest rate
500	5.2
600	4.3
700	3.5

As you specify values in each row, Case Manager Builder by default checks that the values that you enter for a given condition do not overlap or are not identical. For example, consider a column for the age of the customer:

- Row 1: age is between 17 and 30
- Row 2: age is between 29 and 40

In this example, customers that are 29 years old satisfy the condition of both rows. Case Manager Builder reports this as an overlap warning.

By default, Case Manager Builder also checks that the cells in a condition column consider all possible cases, which helps you ensure that there are no gaps in your table. For example, consider a column for the age of the customer:

- Row 1: age is between 17 and 30
- Row 2: age is between 32 and 40

In this example, customers that are 31 years old are not taken into account. Case Manager Builder reports this as a gap warning.

## Defining preconditions in table-based rules

For table-based rules, you can use preconditions to check incoming data to determine whether the table-based rule can process the information. You can also use preconditions to define variables that can be used in the rule. The scope of these variables is limited to the specific rule in which the variables are defined.

## Checking for null values

To avoid receiving possible errors during run time, use defensive programming techniques to ensure that values are populated for all case properties whose values are retrieved in a business rule.

Use the definitions section in a rule to declare variables for all case properties whose values are retrieved. If a case property is not initialized, its variable is not initialized and the rule is not run.

```
definitions
  set balanceAmount to the Opening_Balance of Account_Request;
```

```
if balanceAmount is more than 10000
then set the Interest_Rate of Account_Request to 0.55;
```

## Defining temporary variables

You can define temporary variable in the definitions section to calculate intermediary values that can be used in a rule. For example, you define a business rule to set the value of the ProfitAfterTax case property. Because no case property corresponds to the profits before tax, you define an intermediary ProfitBeforeTax variable to calculate the profits before tax according to the values of the Assets and Liabilities case properties. To determine the correct tax rate, you add a condition in the rule to use a higher tax rate if the profits are more than 1 million dollars.

```
definitions
  set ProfitBeforeTax to
    (the Assets of TaxCasetype - the Liabilities of TaxCasetype);
  set Tax to .07;
  set SurchargeForHigherIncome to .005;

  if ProfitBeforeTax is more than 1000000
  then set the ProfitAfterTax of TaxCaseType to
    (ProfitBeforeTax - ProfitBeforeTax *(Tax+SurchargeForHigherIncome)) ;
  else set the ProfitAfterTax of TaxCaseType to
    (ProfitBeforeTax -ProfitBeforeTax*Tax);
```

## Using Boolean properties in business rules

The following examples show how you include Boolean properties in a rule.

- If the value of a Boolean property is true, set the value of another Boolean property to false:

```
if CaseTypeName is BooleanProperty1
then make it false that CaseTypeName is BooleanProperty2
```




- If the value of a Boolean property is false, set the value of another Boolean property to true:

```
if it is not true that CaseTypeName is BooleanProperty1
then make it true that CaseTypeName is BooleanProperty2
```

For more information about defining rules, see the Business Rules Embedded documentation.

“Setting completion menu options”


### Related concepts:

-  [Designing your case management solution](#)
-  [Business Rules Embedded documentation: Action rules](#)
-  [Business Rules Embedded documentation: Decision tables](#)

### Related tasks:

- “Adding a step to a task” on page 51
- “Adding data fields that can receive values after a step is completed” on page 57
- “Mapping custom parameters in rule steps to external data sources” on page 59

### Related reference:

-  [Sample business rules](#)

## Setting completion menu options

You can set options for the way the rule editor presents terms and phrases in the completion menu, and for the way rules are parsed.

## Procedure

To set completion menu options:

1. In the toolbar for the editing area, click the **Completion Menu Options** icon.
2. Select the check boxes for the options that you want to set:
  - **Enable on Spacebar:**
    - Select to open the completion menu whenever you press **Spacebar** while you type in the editing area.
    - Clear this check box if you want the completion menu to open only when you press **Ctrl+Spacebar**.
  - **Enable on double-click:**
    - Select to open the completion menu when you double-click a word in the editing area.
    - Clear this check box if you want double-clicking to select the current word instead.
  - **Enable auto-restart:**
    - Select to automatically reopen the completion menu when a term or phrase is selected.
    - Clear this check box if you want to open the completion menu by pressing **Ctrl+Spacebar**.
  - **Enable smart mode:**
    - Select to filter the completion menu entries in a smart way to reduce the number of entries.
  - **Enable template mode:**
    - Select if you want the editor to insert relevant placeholders along with the phrases that you select from the completion menu. You complete phrases by clicking the placeholders and selecting options from the completion menu. Enable this mode if you want to insert longer phrases, and then substitute placeholders.
    - Clear this check box if you want the editor to insert only the section of the suggested phrase up to (but not including) the next placeholder. Deactivate this mode if you prefer to build a complete rule in the same way that you might compose an email message.
  - **Use Hierarchical View:**
    - Select if you want the completion menu to group terms and phrases by type, such as Boolean expressions. Grouping items by type can make it easier to navigate long lists of options.
    - Clear this check box if you want the completion menu to list terms and phrases in alphabetical order.
  - **Filter unreachable phrases:**
    - Select if you want the completion menu to hide phrases that cannot be used because there is no suitable rule set parameter or rule variable.
  - **Display toolbar:**
    - Select if you want to display the completion menu toolbar above the list of available terms and phrases.
    - Clear this check box if you want to hide the completion menu toolbar.
  - **Display documentation:**

- Select if you want to display hover help next to the completion menu. When you hover your mouse pointer over a term or phrase, context-sensitive help is displayed in an adjacent panel.
  - Clear this check box if you want to hide the hover help.
3. Click outside the Completion Menu Options window to save your settings.

## Adding tasks

A task consists of one or more steps, a reused business process workflow, or a container task. A case type can have one or more tasks that must be completed by a case worker in the Case Manager Client to close the case.

### About this task

Instead of adding a new task, you can create a task from a business process workflow if your administrator configured access to a business process management system.

Tasks are shown in two views on the Tasks page: priority and sets. In the priority view, the tasks are displayed with required tasks first, optional tasks are next, and discretionary tasks are last. In the set view, the tasks are displayed by whether tasks are mutually exclusive, all-inclusive, or neither.

### Procedure

To add a task:

1. After you create a case type, open the Tasks page and click **Add Task** to choose the type of task to add.

Task type	Description
<b>Task with New FileNet P8 Process</b>	Adds a workflow task for which you define the process in FileNet P8 Process Designer. For information about defining the process, see "Building new FileNet P8 processes to complete tasks" on page 45.
<b>To-do Task</b>	Adds a task that provides a checklist of activities that must be done or information that must be collected for a case type. A to-do task does not have an associated workflow.
<b>Container Task</b>	Adds a container in which you then add subtasks. The subtasks can be workflow tasks, to-do tasks, or even other container tasks.
<b>Task with Existing FileNet P8 process</b>	Adds a task based on an existing FileNet P8 process. For information about creating these processes and making them available for reuse in Case Manager Builder, see "Adding an existing FileNet P8 process as a task" on page 68.
<b>Task with Existing Process</b>	Adds a task that is based on an existing IBM Business Process Manager process. For more information, see "Adding an existing process as a task" on page 70.

1

2. On the General page, specify a name and unique identifier for the task. The unique identifier is added to the name that you entered for the task.

The name of a task must conform to the following rules:

- Begin with an alphabetic character
- Can contain letters, digits, underscores, or spaces
- Can have up to 64 characters although character limits might vary depending on your language
- Not use F\_ or two tilde (~) characters as the first two characters
- Be unique among workflow definitions in the production environment
- Not use the same name as a role in the same solution

3. On the General page, define the behavior of the task:

- a. Define how the task will start by specifying **Automatically**, **Manually**, or **Discretionally**. If you select **Discretionally**, the task must be added to a case programmatically or by the case worker in Case Manager Client.
- b. Optional: Add the task to a set. In the Add Task window, you can add the task only to an existing set. You can create new sets in the Manage Sets window or drag tasks around to different sets in the Tasks page.  
Tasks can belong to a mutually exclusive set, an all-inclusive set, or no set.

#### **Mutually exclusive set**

If the task is part of a mutually exclusive set, the user can complete only one of the tasks in that set.

#### **All-inclusive set**

If the task is part of an all-inclusive set, the user must complete all the tasks in that set.

Tasks that belong to a set must have preconditions.

4. On the Preconditions page, define any preconditions that are required before the task can start. You can have no preconditions, or you can have the task start only if one of the following preconditions are met:
  - When one or more documents are added to the case
  - When a condition is met on a property
  - When a property is modified

You can also define a property expression for any of the precondition types.

If you want the task to start when one or more documents is added, select **A document is filed in the case** in the case precondition and specify an additional property expression. For example, you might require that a loan application document is added to the case before the loan application task can start.

5. On the Task Properties page, add and configure the properties that are associated with this task.

Task properties provide a mechanism for tracking task-specific information beyond the of the task workflow. For example, you might have a repeating task to process witness statements. Each instance of a witness statement task has properties to record information such as who processed the witness statement and the date that the statement was taken. This information can be maintained through the task properties rather than through case properties.

For to-do tasks, which are displayed as checklists, all associated task properties are shown as to-do items in the default view.

You can create properties for a task or reuse existing properties from the solution or object store. When you create a task property, that property

becomes one of the solution properties and can be used as a property for a case type or other tasks. You can also use a task property in an in-basket.

6. Optional: On the Design Comment page, add a design comment that explains, for example, the reasons that this task was created, or what the imported process does, or how it works with the solution.
7. Click **OK**.

## What to do next

To create steps for a task, click the **Edit steps** icon. To customize the view for a to-do task, click the **Open To-do View Designer** icon.

“Tasks in solutions”

“Task preconditions” on page 43

“Task initiation” on page 44

“Task states” on page 45

“Building new FileNet P8 processes to complete tasks” on page 45

“Integrating business processes with IBM Case Manager solutions” on page 67

“Designing views for to-do tasks” on page 73

### Related tasks:

“Integrating business processes with IBM Case Manager solutions” on page 67

## Tasks in solutions

A *task* represents a specific activity that is performed as part of a case. A task can consist of one or more steps that must be completed to complete the task.

Alternatively, a task can be a simple checklist item that is not associated with a workflow.

“Workflow tasks”

“Custom workflow tasks” on page 42

“To-do tasks” on page 42

“Quick tasks” on page 43

### Workflow tasks:

A *workflow task* represents a process that is performed regularly for a case. As part of the case type, you define the workflow for the task. This workflow defines the sequence of steps or work items that must be completed for the task along with the role or user that must complete each step. You can define the process that is associated with the workflow task in either IBM Business Process Manager Process Designer or FileNet P8 Process Designer.

You specify that a workflow task is required for the case to complete or that it is optional. You also specify whether the task is started manually, automatically, or when a user adds the task in the Case Manager Client.

You can hide tasks from view in the case client. For example, if you have tasks that have no value to the case worker, you can mark those tasks as hidden when you design your solution. Typically hidden tasks are controlled programmatically through business logic that is built into the case management application.

You can mark a task to repeat when there is a property change or when there is a document filing precondition defined for the task. A task that is marked as repeatable can occur multiple times during the lifetime of the case in which it

resides and can cause new tasks to be created and repeated. Task types can be restarted as needed to repeat, even if the task is already in complete state.

You can group predefined tasks by sets: all-inclusive or mutually exclusive. Adding tasks to an all-inclusive set means that all tasks in that set must be completed. Adding tasks to a mutually exclusive set means that if you start one task in the set, you cannot start any of the others. A task can be included in only one set type. A task that is included in a set must have a precondition.

To simplify a workflow, you can break a large task into smaller, more manageable tasks. For example, one task might be to process a loan request. You might break this large task into separate tasks for assembling the loan documents, reviewing the documents, and accepting or rejecting the request.

Generally, a case is not complete until all required tasks are completed or manually disabled, or any running task is completed or canceled. However, you can specify that some tasks don't affect whether a case is considered complete, and are automatically stopped when the case completes. For example, you can use this setting for a task that generates daily reports. When the case is completed, the reports are no longer needed and the task stops automatically.

**Important:** You cannot select the **Stopped when the case completes and does not affect case completion** setting for all tasks in the case. If this option is selected for all tasks in the case, the case never completes.

### **Required tasks**

Workflow tasks that you make required for the case can be started automatically or manually as soon as the case is created or after preconditions are met for the task. For example, if the solution that you are designing is for credit card disputes and one of the case types is for claims with supporting documentation, you can create a required task for a claim review as soon as supporting documentation for a claim is added to the repository.

### **Optional tasks**

Workflow tasks that make optional for the case can be started automatically, or manually as soon as the case is created or after preconditions are met for the task. For example, if the solution that you are designing is for automobile claims and one of the case types is for automobile accidents, you can create an optional task that can be manually started for the rental car task. Optional tasks are displayed second on the Tasks page.

### **Discretionary tasks**

Workflow tasks that you make discretionary can be added by case workers as needed after a case is created. For example, if the solution that you are designing is for automobile claims and one of the case types is for automobile accidents, you can create a discretionary task to request that a field agent investigate the accident that is submitted by an insurance adjuster for possible fraud. Discretionary tasks are displayed last in the Tasks page.

### **Container tasks**

You can define workflow tasks that contain other tasks, which are called subtasks. Container tasks can start automatically, manually, or discretionary. Subtasks are

created after the container task starts. The subtasks within a container task cannot be discretionary. They can be only automatic or manual.

When you design a solution, you can create subtasks that start after the container task starts and the preconditions for the subtasks are met. For example, after a loan reaches the approved state, a bank might define two subtasks:

- One subtask for notifying the customer of the loan approval.
- One subtask for providing the funds to the loan applicant.

In this example, the loan process task is the container task that has the precondition of `loanstatus=approved`. The customer notification and funding tasks are subtasks that start only when the case property "loan status" for the loan case equals "approved".

If your administrator configured access to a business process management system, you can place FileNet Business Process Manager and IBM Business Process Manager Advanced subtasks inside of container tasks.

### **Subtasks**

The subtasks that you create within a container task are created in Case Manager Client only after the container task that they are in moves to working state. You cannot mark subtasks as discretionary. However, you can move subtasks out of a container task, and then mark them as discretionary.

### **Custom workflow tasks:**

You can enable case workers to create custom tasks for a case type. Case workers can create custom workflow tasks to address unexpected or one-time activities that are related to a case, but which are not formally defined as part of the solution. For example, a case worker determines that a document needs to be reviewed by a person who is not typically part of the review team for a case.

To define the steps that are required to complete a custom workflow task, Case workers use the custom task editor in Case Manager Client.

### **To-do tasks:**

A *to-do* task represents a simple activity that does not have workflow steps. To-do tasks provide a checklist of activities that must be done or information that must be collected for a case type. For example, if the solution that you are designing is for automobile claims and one of the case types is for automobile accidents, you can create a to-do task to enter the completion of making contact by telephone with a claimant.

You specify that a to-do task is required for the case to complete or that it is optional. You also specify whether the task is started automatically or when a user adds the task in the Case Manager Client.

You can hide tasks from view in the case client. For example, if you have tasks that have no value to the case worker, you can mark those tasks as hidden when you design your solution. Typically hidden tasks are controlled programmatically through business logic that is built into the case management application.

To-do tasks are displayed in the To-Do List widget.



## Quick tasks:

A *quick task* represents a one-time activity that is related to a case but not associated with a workflow. You can enable case workers to create quick tasks for a case type to help them organize their work or address unexpected or one-time activities. For example, a case worker determines that a follow-up call to a customer is required and adds a quick task to track this requirement. The case worker can assign the quick tasks to another case worker and set a due date.

Quick tasks are displayed in the To-Do List widget. A case worker can create a quick task by just typing in the name. After the task is created, the case worker can add a description, assign the task, and set the due date. A quick task is marked closed by a single click in the To-Do List widget.

## Task preconditions

For manual and automatic tasks, you can specify preconditions that must be met before the task is ready to start. Tasks can start automatically after all preconditions are met or manually by a user after all preconditions are met.

If the preconditions for a task are not fully satisfied and if the task has a trigger, such as a case property is updated or a document is filed, the task remains in waiting state until the next trigger occurs to reevaluate the preconditions.

A task with preconditions is complete after all of the preconditions are met, but will not start until the trigger you defined has all of the preconditions met, such as when a second document is added.

There are four types of preconditions for a task:

### No precondition, start task

A task can have no preconditions, that is, no precondition must be met for this task to start.

### A property condition is met

You can design a task that starts only if the conditions are met. You specify the conditions by building expressions that contain case properties. For example, you can build a property expression that contains the following conditions: one of the case properties must equal a value, a second property value must be true, and a third property must begin with a value before the task can begin.

You can join the conditions by using the Boolean ALL or ANY operators.

The operators in the expression change depending on the property type.

### A case property is updated

You can design a task to start only when one or more case properties are updated.

Optionally, you can specify additional property conditions. If you specify additional property conditions, the property conditions must also be met for a task to start.

You can also mark the task as repeatable when the precondition is satisfied.

### A document is filed in the case

You can design a task to start only when documents of one or more


document classes from the choice list are added to the case folder. Or you can design a task to start when documents of any document class are added to the case folder.

Optionally, you can specify additional property conditions. If you specify additional property conditions, the conditions must also be met for a task to start.

You can mark the task as repeatable, to repeat when the precondition is satisfied. For example, if your task is to review an automobile accident claim estimate, you can define the task is repeatable. A new task will be created and work added to the in-basket of the case worker that reviews the claim estimate each time that a repair estimate document is added to the case.

You can update the preconditions for a task at any time, including on deployed solutions. If you change the expression of a property condition for a deployed solution, for task types that have the A property condition is met precondition, run the precondition checker utility after you redeploy the solution.

**Related reference:**

 [Validating preconditions](#)

## **Task initiation**

A case type can contain many tasks, and each task can be started by using one of three methods. The decisions that you make when you design a task affect what a case worker can view and edit in Case Manager Client when the task starts.

You can define the workflow for a task by using the Step Designer in Case Manager Builder. In the Step Designer, you select the case properties and task properties that are used as parameters for each step. The parameters determine what information the case worker can view and edit for each step. The Step Designer automatically generates the values to be assigned to the step parameters after the processing of the launch step completes. Case Manager Client uses these assigned values to synchronize the case properties and task properties with the step parameters.

The mapping of case properties or task properties to workflow fields determines the initial values of the step parameters when a task is started. When a task is started, the step parameters are initialized with the property values.

When a case worker opens a work item, the Work Details page opens or displays. Case Manager Client populates the fields with the property values from the case instance. If a property value is null, Case Manager Client populates the field with the appropriate default value. For example, a null integer property is populated with a 0.

## **Tasks that start automatically**

You can define a task that starts automatically. Optionally, you can specify that the task starts based on a precondition such as when an estimation document is added to an insurance claim case. If you do not specify a precondition, the task starts automatically when the case is created.

In Case Manager Client, a task that starts automatically does not open a Work Details page, so the case worker cannot see the work item for the Launch step. Instead, the default values that were set for the properties are used to start the

task. When the task is started, or when a case is added and the automatic tasks start, the user will see the work items in their in-basket.

### **Tasks that start manually**

You can define a task that a case worker starts manually. You can define preconditions that must be met to put the task into Ready state. However, the task does not start until the case worker decides to manually start the task.

In Case Manager Client, the user starts the task by selecting the task in the Case Information widget on the Case Details page and clicking **Start**. The default values that were set for the properties are used. The case worker cannot edit the property values until the task progresses to a step where the properties are available.

### **Tasks that start when they are added to the workflow at runtime**

You can define a discretionary task that starts when a case worker or the system creates an instance of a task. As part of defining the workflow for this task, you specify parameters, such as policy number and accident details, for the launch step that map to the case properties.

In Case Manager Client, a case worker can add a discretionary task to the Case Details page. The launch parameters that you defined for the task are then displayed in the Add Task page for the case worker to edit. The system then updates the corresponding case properties with the parameter values.

### **Custom tasks**

You can configure a case type to allow a case worker to create a custom task at run time.

In Case Manager Client, the case worker defines and starts a custom task whenever one is needed.

### **Task states**

Tasks have different states that are displayed to the case worker in the Case Manager Client. For example, a manual task that has met all of its preconditions is in the **Ready** state. An automatic task that has met all of its preconditions is in the **Started** state. Depending on how the task is defined it might require to be started manually by the case worker.

In addition to task state, there is a case state. When all required tasks are completed and there are no more running tasks, the case is marked complete. For example, after a task is started, it is in **Working** state and the task must be completed for the case to complete, even if the task was initially set as optional.

### **Building new FileNet P8 processes to complete tasks**

You can use FileNet P8 Process Designer to define and connect steps in a process that case workers must complete so that a workflow task can be completed. The process automates the routing and processing of your case documents and case data for a specific business process.

### **About this task**

You can build a FileNet P8 process that consists of the following information by using the Step Designer in Case Manager Builder:

## Graphical map

The graphical map shows the sequence of steps that must be completed for the business process. Each step represents a specific action in the business process.

## Swimlanes

Each task includes a System swimlane and a Launch Step swimlane. The System swimlane contains rule steps and system steps that are assigned to FileNet system processes, such as component steps and submap steps that can be created with Process Designer. Steps in the System swimlane cannot be edited in the Step Designer. Placeholder steps, stub steps, can be added to the system swimlane for component, submap, or systems steps. You cannot add steps to the System swimlane and you cannot delete the System swimlane.

Each workflow contains a Launch Step swimlane. For tasks that start automatically or manually, you cannot edit the steps in the Launch Step swimlane. For discretionary tasks, you can edit the steps in the Launch Step swimlane. For example, you can assign a workgroup to the Launch Step for the case worker to assign the workgroup members. You cannot add steps to the Launch Step swimlane. Supported steps that were added to the workflow by Process Designer that do not have a workgroup or role associated are shown in the Launch Step swimlane. You cannot delete the Launch Step swimlane.

**Steps** To create a step you must add it to a swimlane. Connectors set the routing between steps in the swimlanes. Connectors also have properties such as name, description, and condition. The step response can be used as a connector condition.

After a step is initially assigned to a swimlane, you can move it to another swimlane only by updating the **Swimlane** step property. A step that you move to a different swimlane retains all connectors.

You add details for each step, including who completes the step, which attachments are required, what data is necessary, what responses the participant can choose, deadlines, and other step properties.

You can add a stage step to move the case to handle the lifecycle of case stages. A stage step can be used to move the case to the next stage, put the current stage on hold, release the hold on the current stage, or restart the previous stage.

You can add a rule step to the System swimlane to determine process routing or update case properties based on a business rule. Before you create a rule step, you must define the business rule that you want to associate with the rule step.

You can add a property step to the System swimlane to set the value of a case or task property. For example, you want to offer a 20% discount when the transaction amount is more than \$1,000. You create a task and specify a precondition that the task runs only if the Transaction\_Amount property is more than 1000. In the task, you add a property step to set the Product\_Discount property to 0.2. Alternatively, you can use the property step to set the Product\_Discount property to the value of another property, such as Discount\_Allowed.

You can add a placeholder step, or stub step, to the System swimlane for component, submap, or system steps.

To add data fields to a step, use Process Designer.

## Connectors

Routing logic specifies how work advances from one step to the next. Routing can branch based on the user responses that are defined for the step or the routing might be straight from one step to the next. Routes can cross swimlanes to permit workers in different roles or workgroups to complete the steps.

## Procedure

To build a workflow by using FileNet P8 Process Designer:

1. From the Tasks page, click the **Edit steps** icon to display the step designer for a task.
2. Optional: Click **Manage Workgroups** to add a workgroup that can include users or groups. You must create a workgroup before creating a workgroup swimlane. A case worker can specify the workgroup members later.
3. From the **Palette**, add swimlanes.
  - To create a swimlane for a specific role that is defined in this solution, drag **Role Lane** to the canvas.
  - To create a swimlane for a specific workgroup that is defined in this task, drag **Workgroup Lane** to the canvas.

By default, roles and workgroups are assigned to swimlanes in alphabetical order. You can change the role or workgroup for a swimlane in the Role Property or Workgroup Property section.

4. To create a new step, drag **Step** to a specific swimlane in the canvas.
5. Select the step and specify properties for the step in the **Step Properties** section, then click **OK**. You can specify a step name, description for the step, the instructions that are displayed to the case worker for this step, a deadline for completing the step, case worker responses to the step, and other step properties.

You can also specify the case properties, attachments, or data fields, and the workgroup that the step is assigned to, in the **Step Properties** section.

6. Create routes in the workflow from one step to another. You can define a name and description. Select the appropriate response or no condition.
7. Apply your changes and validate the workflow. You can also specify the case properties, response, attachments, or data fields, and the workgroup that the step is assigned to, in the Step Properties section. The steps and routes are validated and a message is displayed in the status bar. Correct any errors before proceeding.

**Important:** The validation tool performs syntactic validation only.

8. To save the workflow, the task, and the solution, click **Save**.

**Attention:** If you add a swimlane, but you do not add any steps to the swimlane, the swimlane will not be saved when you save the workflow.

“Adding a workgroup to a task” on page 48

“Adding attachments to a task” on page 50

“Adding a swimlane to a task” on page 50

“Adding a step to a task” on page 51

“Creating routes in a workflow” on page 60

“Adding workflows that are not associated with tasks” on page 66

## Related concepts:

“Route validation rules for a workflow” on page 61

“Business rules” on page 33

**Related tasks:**

“Adding an existing FileNet P8 process as a task” on page 68

“Adding an existing process as a task” on page 70

**Adding a workgroup to a task:**

Workgroups provide a way to assign work to particular users. A case worker defines the users or groups in the workgroup in the Case Manager Client.

**About this task**

Before you add a step that must be completed by members of a specific workgroup, you must have a step for a case worker to add users to the workgroup. You can create a separate step or you can add the action to edit the workgroup to the Launch step for the task. The task must be a discretionary task or else you cannot edit the Launch step. You can also use Process Designer instead of Step Designer to add a workgroup to a step.

**Procedure**

To add a workgroup to a task:

1. Click **Manage Workgroups**, then click **Add Workgroup**.
2. Provide a prompt for this workgroup. A prompt is displayed in Case Manager Client to remind the case worker to add members to the workgroup.
3. Click **OK**.
4. Click **Close**.
5. Identify the step in your process that must be assigned to a workgroup.
6. Add the step for the workgroup to complete:
  - a. Add a workgroup lane for the new step. If there is more than one workgroup defined, change the workgroup in the Workgroup Property section.
  - b. Add a step to the lane.
  - c. In the Step Properties section, set the properties as needed for this step.
  - d. Click **OK**. Then, click **Close**.
7. Add a connector from the **LaunchStep** or the other step where the workgroup members are assigned to the step that the workgroup must complete.
8. Apply your changes and validate the workflow.

“Adding a step to assign workgroup members”  
“Assigning members to a workgroup in the Launch step” on page 49

*Adding a step to assign workgroup members:*

If you include a workgroup in a task, create a step for the case worker to assign members to the workgroup.

**Before you begin**

Before you add a step that must be completed by members of a specific workgroup, you must have a step for a case worker to assign users to the workgroup.

## Procedure

To add a step to assign workgroup members:

1. Add a lane for the new step. Do not create a workgroup lane for the workgroup that must be assigned. This step must be completed by users who are assigned to a role or by users who are assigned to a workgroup that already has members. That is, there must be a known user who can complete this step. The case worker is prompted to assign a user or group to the workgroup when the work item is opened from the in-basket.
2. Add a step to the lane.
3. In the Step Properties section, select a workgroup from the list.
4. Ensure that **Read and Write** is selected.
5. Click **OK**.

## Results

The case worker is prompted to assign a user or group to the workgroup when the task starts.

*Assigning members to a workgroup in the Launch step:*

If you include a workgroup in a task, you can have the case worker assign members to the workgroup in the Launch step.

## About this task

Before you add a step that must be completed by members of a specific workgroup, you must have a step for a case worker to add users to the workgroup. You can add the action to edit the workgroup to the Launch step for the task.

**Restriction:** You can edit the Launch step properties only for a discretionary task.

## Procedure

To assign members to a workgroup in the Launch step:

1. Click **Launchstep** in the **Launch Step** lane.
2. In the Step Properties section, select a workgroup from the list.
3. Ensure that **Read and Write** is selected.
4. Click **OK**.

## Results

The case worker is prompted to assign a user or group to the workgroup when the task starts.

## What to do next

You can also use Process Designer to display the workgroup field in the Launch Step in your case management application.

### **Adding attachments to a task:**

You can add attachments to tasks for the documents that a case worker needs for completing a task. For example, a case worker might need a claim application document, a policy document, and damage photographs to process an insurance claim.

#### **About this task**

In Case Manager Builder, an attachment is the designated location where users who work in Case Manager Client can attach documents.

For example, for the case worker to view the claim application document, a policy document, and damage photographs, each of these document types must be added to the task as an attachment. The claim application document attachment and the damage photographs documents might be set to read and write so that the case worker can update them, while the policy document attachment is set to read only.

If you define a precondition to start the task when a new document is added to the repository, you can designate that document as the **Initiating Attachment** as described in the following steps. If an initiating document is specified, then the task starts when that document is received and the corresponding attachment field is automatically set to that document. If no initiating attachment is specified, then the precondition will still start the task, but the precondition document value will not be set in the attachment field.

#### **Procedure**

To add an attachment to a task:

1. Click the Attachments button.
2. Click **Add Attachment**.
3. Enter a name and prompt for the attachment. The prompt is displayed in the Case Manager Client to provide a brief hint to the case worker about the document.
4. Click **OK**.
5. Optional: If a document precondition is set, select the attachment from the **Initiating Attachment** dropdown menu.
6. Click **OK** and then click **Close**.

#### **Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

### **Adding a swimlane to a task:**

You add swimlanes to a task to divide the work among roles and workgroups. You add steps to swimlanes.

#### **Before you begin**

Before you can add a workgroup swimlane you must create one or more workgroups.



## Procedure

Drag a role or workgroup swimlane to the canvas. A new swimlane automatically takes the name of an existing role or workgroup. You can change the role or workgroup by editing the swimlane properties.

## Results

You can add steps to the new swimlane. If you close the Step Designer before you add steps to the swimlane, the empty swimlane is not saved in the solution.

“Swimlanes”

### Related tasks:

“Adding a workgroup to a task” on page 48

*Swimlanes:*

You can divide the work that is required to complete a task among different roles or workgroups by creating swimlanes.

A *swimlane* is a partition in the Step Designer canvas that organizes the role or workgroup responsible for a step. You can drag and drop steps only to a role or workgroup swimlane in the canvas. After a step is first assigned to a swimlane, you can move it to another swimlane only by updating the **Swimlane** step property.

You can add two types of swimlanes to the canvas: role lanes and workgroup lanes. You can assign any role that is defined in the solution to a role lane. Any steps added to a role lane are assigned to case workers in that role. You can add workgroups in **Manage Workgroups**. Any steps that are added to a workgroup lane will be in the personal in-basket for users that are assigned to that workgroup.

The following table describes the differences between a role and a workgroup.

*Table 7. Differences between roles and workgroups*

<b>Roles</b>	<b>Workgroups</b>
Roles are created at the solution level and can be reused for more than one task or case type.	Workgroups can be created at the task level. You can also select predetermined workgroups that you create in Case Manager Builder or in Process Designer.
Each role is assigned an in-basket at the solution level.	Workgroups do not have in-baskets. Work assigned to workgroup members is located in the personal in-basket for the member.
The Process Engine creates a work queue for each role in the solution.	Workgroups are not assigned to a queue. Each member of the workgroup receives the work item. When a work item (step) is assigned to a workgroup to complete, all members of the work group must process and complete the item.

### Adding a step to a task:

You can add steps or stub steps that must be completed to complete the task. You can add a rule step to determine process routing or update case properties. You

can add a property step to set the value of a case or task property. You can add a stage step to move the current stage to another state.

### **Before you begin**

Before you can add a step to a task, you must add one or more swimlanes. Before you can add a rule step, you must define the business rule that you want to associate with the rule step. Before you can add a stage step, you must define the stages for the case type.

### **Procedure**

To add a step to a task:

1. Drag a step to a swimlane in the canvas. You add a regular step to a role or workgroup swimlane. You add a property step, a rule step, a stage step, or a stub step to the System swimlane.

2. Specify the step properties.

For a regular step, you specify properties, such as name, description, deadline, and workgroups.

For a property step, select the property that is to be updated from the **Property** list, then specify how that property is to be set.

For a rule step, edit the **Rule Name** property to associate the rule step with a business rule.

For a stage step, select one of the following values from the **Action** list to change the state of the current stage:

#### **Advance**

Completes the current stage and moves the case to the next stage.

#### **Go to Previous**

Returns the case to the previous stage.

#### **Put on Hold**

Puts the current stage on hold.

#### **Release**

Removes the hold on the current stage.

“Steps” on page 53

“Assigning properties to a step” on page 56

“Adding steps that use case properties in Process Designer” on page 57

“Adding data fields that can receive values after a step is completed” on page 57

“Adding steps in Process Designer to integrate into tasks” on page 58

“Mapping custom parameters in rule steps to external data sources” on page 59

“Adding a custom page to a step” on page 60

### **Related concepts:**

“Business rules” on page 33

### **Related tasks:**

“Adding a swimlane to a task” on page 50

### *Steps:*

*Steps* are the actions in a workflow that must be completed for a task. You can create and edit steps, assign the steps to be completed by a role or workgroup, and connect steps into a workflow.

### **Regular steps**

Case workers see steps as work items in the Case Manager Client.

For each step, you can define properties such as:

- A description of the work item.
- Instructional text for the case worker to complete the work item. Case workers can click **View Instructions** in their toolbar to view these instructions.
- Responses that the case worker can choose for the work item. For example, you can define responses such as approve or reject.
- Whether the work item can be reassigned by a case worker to another case worker. If you select **false**, the case worker cannot reassign the step.
- A deadline for completing the work item.

A step can include an event with a condition, a deadline, or an action that a user with a specific role must complete. A step can depend on the completion of prior or concurrent steps.

A step can be required. The action resulting from a step can be a process, a notification, or other action. An action can have a deadline with different results if the deadline is met or not met.

You can also set parameters in the Step Properties section for case properties, workgroups, attachments, and data fields.

### **Case Property**

For each step, you can assign case properties that can be read-only or can be viewed and modified. You cannot add new case properties from the Step Designer. You can assign only properties that are added for this case type.

### **Workgroup**

For each step, you can expose the workgroup field to the step so that case worker can assign or remove more users or groups to the workgroup. Add new workgroups by clicking **Manage Workgroups**.

Workgroups provide a way to assign work to particular users instead of to any user in a role. The users or groups in the workgroup are defined in the Case Manager Client. When users are processing a work item, they see these workgroups in the Properties widget and are able to select users and groups to assign to the workgroup. For example, you might have a workgroup named Reviewers, but you do not know who the reviewers are when you are designing the solution. Instead, it is a decision made by the case worker in the Case Manager Client. When the case worker views the case in the Properties widget, he or she can assign users to the Reviewers group.

### **Attachment**

Select which document is available for a case worker when processing the step. Add new attachments by clicking **Manage Attachments**.

You specify only a name for the attachment that acts as a place holder for an array of attachments. The actual attachment is defined as a workflow property in Process Designer.

### **Data Field**

Properties created in the Process Designer. These properties are read only.

### **Expressions**

You can build expressions to evaluate conditions for routing decisions in Process Designer. The Step Designer only creates and edits routing based on responses. You can also use expressions to update property values.

You can use Process Designer's Expression builder to define case properties in workflow expressions such as step parameters, route conditions, or assignments. For example, you can specify `F_CaseFolder.SolutionPrefix_AccountNumber` for a step parameter expression.

### **Property steps**

You can add a property step to set the value of a case or task property. For example, you want to offer a 20% discount when the transaction amount is more than \$1,000. You create a task and specify a precondition that the task runs only if the `Transaction_Amount` property is more than 1000. In the task, you add a property step to set the `Product_Discount` property to 0.2. Alternatively, you can use the property step to set the `Product_Discount` property to the value of another property, such as `Discount_Allowed`.

### **Rule steps**

You can add a rule step to determine process routing or update case properties based on a business rule. Before you create a rule step, you must define the business rule that you want to associate with the rule step.

### **Stage steps**

You can add a stage step to the case to handle the lifecycle of case stages. A stage step can be used to move the case to the next stage, put the current stage on hold, release the hold on the current stage, or restart the previous stage.

### **Steps that you can add by using Process Designer**

Depending on your business requirements, you can add more complex steps. You can add system steps, task steps, submap steps, or component steps by using Process Designer.

#### **System steps**

You can create system steps and edit the properties for system steps in Process Designer. These steps include one or more FileNet system functions. System steps are displayed as read-only steps in Step Designer.

#### **Task steps**

You can create task steps and edit the properties for task steps in Process Designer. When you create the step in the Process Designer, you must assign the steps to the step processor for your solution. For example, select the `solution_prefix_CmACMStep_DEFAULT_PAGE` processor.

**Tip:** To quickly add a step, copy an existing step that was created in Case Manager Builder and change the step name and the user or queue that the step is assigned to. You can also use the Business Objects function in Process Designer to add the case properties to the step in Process Designer.

### Submap steps

You can create and edit submaps in Process Designer. A submap step represents a call from the current workflow map to another map in the same workflow definition. Submap steps are displayed as read-only steps in Step Designer.

When you create steps in the submap in the Process Designer, you must assign the steps to the step processor for your solution. For example, select the `solution_prefix_CmACMSTEP_DEFAULT_PAGE` processor.

**Tip:** To quickly add a step to a submap, copy an existing step that was created in Case Manager Builder to the sub map and change the step name and the user or queue that the step is assigned to. You can also use the Business Object functionality in Process Designer.

### Component steps

You can create component steps and edit the properties for component steps in Process Designer. These steps route work to operations in custom Java™ or Java Message Service (JMS) components. Component steps are displayed as read-only steps in Step Designer.

When a solution is edited in the Process Designer, resources from the isolated region are not available for use because the region might not be created yet in the IBM Case Manager offline template model. If you want to refer to the CE\_Operation component from a component queue step in a workflow definition originating from Case Manager Builder, register the CE\_Operations component queue first.

**Remember:** Designing solutions is done offline and you will not see any component queues, including the CE\_Operations component queue.

### Optional step properties

Depending on your business requirements, you can specify optional properties for a step.

#### Data fields

You can create and edit data fields in Process Designer. Data fields that are exposed on a step are displayed as read-only items for the **Parameters** property for a step in Step Designer.

#### Attachments

Step Designer supports only attachment arrays. You can add individual attachments by using Process Designer.

#### ToolTips

You can create and edit tool tips for the in-basket filter in Process Designer. However, if you open the solution in Case Manager Builder your tool tips are overwritten.

#### Task independence

Each task is independent of other tasks in Case Manager Builder:

- Tasks do not share case data. The solution properties and document types are templates that can be shared by tasks, but the case data and documents in a running task (case work item) are not inherited or shared with another task.
- The steps in one task cannot wait for steps in a different task to complete.
- A task cannot create another task.

**Related concepts:**

“Roles in solutions” on page 8

“Business rules” on page 33

**Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

*Assigning properties to a step:*

You can assign properties, such as name, responses, or a role, to a step for a case worker to complete.

**About this task**

**Important:** For a case worker to view or edit the value of a property, including properties that are assigned to an in-basket, you must add the property to the case type and to a specific step for the case worker.

If you assigned a default value to a property for a solution or for a case, the default value is used for the step property value when a new case starts. You can assign properties for the Launch Step only if the task is a discretionary task.

**Restriction:** You can specify only name and description for stub steps in Step Designer. Use Process Designer to add other properties to stub steps.

**Procedure**

To assign a property to a step:

1. In the Step Designer, click the step.
2. In the Step Properties section, specify the name of the step, a description, and any instructions for the case worker.
3. Optional: Add a response from the menu. You can add responses by clicking the **Edit** icon (...).
4. Select whether the task can be reassigned to another case worker from the menu.
5. You can specify the deadline and when the reminder is sent in minutes, hours, days, or weeks.
6. Set the read and write option for each case property, workgroup, or attachment. The **Read and Write** option is selected by default, and the case worker can edit the property value. Select **Read only** to prevent the case worker from changing the property value.
7. Select the page layout, a swimlane, and split and join operators. If you need to move a step from one swimlane to another, change the swimlane property.
8. Click **Save**. To save the changes that you made to the step properties, ensure that you click **Save**. Any unsaved changes will be lost when Step Designer is closed.

*Adding steps that use case properties in Process Designer:*

You can map step properties to a case folder instead of to a workflow.

### **About this task**

Step properties that are added to a workflow step in Case Manager Builder map to the case folder. Step properties that are added to a workflow step in Process Designer map to the workflow. You can map step properties to the case folder instead of to the workflow in Process Designer, so that the step properties are displayed for the step in the work item in Case Manager Client. To map step properties to a case folder, use the Process Designer Business Object functionality, which you can access in the step Parameters tab.

### **Procedure**

To map step properties to a case folder in Process Designer so that they display properly in Case Manager Client:

1. In the Case Manager Builder Tasks page, click the Open Process Designer icon, which is displayed next to the task. You can also open Process Designer from the **More** menu of the solution in the Manage Solutions page.
2. Select the case type, and then open the task. If your task is not displayed, click **View Workflows**.
3. Add a general step and assign the step to a role, workgroup, or participant.
4. Go to **General > Step Processor > case\_prefixCmAcmSTEP\_DEFAULT\_PAGE** to assign the step processor.
5. In the Parameters tab, select the case parameters, and then click the Business Objects... icon in the Selected Parameters window. F\_CaseFolder is already selected.
6. Select the case properties that you want, and then click the arrow icon to move the properties to the Selected Parameters box.
7. In the Selected Parameters box, select all of the properties that you added, and then use the Access Rights icon in the selected fields section to change the access to Read/Write.
8. Validate the workflow.

**Tip:** These property changes might cause errors when you validate the workflow in Step Designer. The errors do not prevent the solution from being deployed.

### **Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

*Adding data fields that can receive values after a step is completed:*

You can add data fields to a step, then assign values to the data fields after a step is completed.

### **Procedure**

To add data fields that can be assigned values after a step is completed:

1. In Process Designer, open your solution.
2. Select the case type.

3. On the Workflow Properties page, select Data Fields and create new data fields for your steps.
4. Select a role step, then click the **Parameters** tab.
5. Select parameters from the **Available Parameter** column and move them to the **Selected Parameters** column.
6. On the Assignments page, select **After Completion**.
7. Add the new data fields to the Field Assignments by selecting a data field in the **Name** column, then select or specify an expression.
8. In Expression Builder, set a value, expression, or function for the field.
9. Select **Functions**, then enter a value, an expression, a function, or another field.
10. Click **Insert**.
11. Click **OK**.
12. Validate the workflow.

### Results

The new data fields are available in the step when opened in Case Manager Builder. When the step completes, the value of the data field is updated.

*Adding steps in Process Designer to integrate into tasks:*

FileNet P8 Process Engine supports extended workflow features that the Case Manager Builder does not support. You can edit your solution workflows in Process Designer, and later continue working in the Step Designer.

### Procedure

To add steps in Process Designer:

1. Open Process Designer by using one of the following methods:
  - From the Manage Solutions page in Case Manager Builder, click **Open Process Designer** from the **More** menu located at the bottom menu of your solution.
  - Edit your solution, go to the Tasks page, and then select **Open Process Designer**, which is located after the **Edit Steps** option.

You can then browse to the solution definition, case type, and workflow task.

**Tip:** If you open Process Designer by clicking its icon on a specific task in the Solution's Task page, you can edit the workflow for that task and other tasks in the case type. However you cannot view Configuration, Roles, or In-baskets. If you open Process Designer at the solution level in Manage Solutions and select the case type, you can add workflows, roles, in-baskets, and other items. In addition, you can open different case types without closing Process Designer.

**Restriction:** Do not open a workflow in Step Designer at the same time that you open the workflow in Process Designer. Never open the PE Configuration file or the XPDL file in another application when the solution is open in Case Manager Builder.

2. Add or edit steps as required for your business requirements.
3. Click **Validate**.
4. To save and close the file, choose **File > Solution > Save and close**.
5. Exit Process Designer.



6. Optional: Review the task in Case Manager Builder.

## Results

**Restriction:** When editing a solution workflow collection, you cannot make changes to queue security or to component queue properties by using the Process Configuration console that is launched from Process Designer. Instead, use Administration Console for Content Platform Engine.

### Related information:

Cannot access Process Designer from Case Manager Builder with Google Chrome

*Mapping custom parameters in rule steps to external data sources:*

If a rule step is associated with a business rule that includes custom rule parameters, you must edit the rule step in Process Designer to map the custom rule parameters to the external data sources.

## Before you begin

For each custom rule parameter, you must define a data field whose value is populated by a previous step in the workflow. For example, if the associated rule uses a CreditRating custom parameter, define a CreditRating data field that is populated after completion of a step that receives credit ratings from a web service. You can then map each custom rule parameter to the appropriate data field.

## Procedure

To map custom rule parameters in a rule step to external data sources:

1. Open Process Designer. On the Case Manager Builder Tasks page, click the **Open Process Designer** icon that is displayed next to the task. Alternatively, you can open Process Designer from the **More** menu of the solution on the Manage Solutions page.
2. Select the rule step that uses custom parameters.
3. From the **Operation Parameters** list, open the expression for the CustomRuleParameterValues parameter in Expression Builder and specify the data fields to map to the custom rule parameters that are used by the rule step. Ensure that you specify the values in the same order as the custom rule parameter names are listed in the CustomRuleParameterNames parameter.

**Tip:** If after you create a rule step you edit the associated rule and add a custom parameter, the custom parameter is not automatically included in the CustomRuleParameterNames parameter value. Manually add the parameter to the list of values.

For multiple-value custom rule parameters, specify the value in the format `arrayToString(data_field,"{","}"",",")`. For example, if your rule step uses the InterestRate and CreditRatings custom rule parameters, and you defined data fields with the same names, specify the following value for the CustomRuleParameterValues parameter:

```
{"InterestRate",arrayToString(CreditRatings,"{","}"",",")}
```

4. Validate the workflow.

### Related concepts:

“Business rules” on page 33

### Related tasks:

“Adding data fields that can receive values after a step is completed” on page 57

*Adding a custom page to a step:*

Instead of displaying a default page to the case worker, you can create custom pages to display in Case Manager Client and assign the custom page to the steps in your workflow.

### **About this task**

By default, each case worker sees the default page in Case Manager Client for working with a step. You can create custom pages to display other information in the Page Designer, and assign the pages to steps.

### **Procedure**

To add a custom page to a step:

1. Create the custom step page.
  - a. On the **Pages** tab, add a page of one of the available page types.
  - b. Click the page name to design and configure the page layout.
  - c. In Page Designer, design the page layout.
2. Open the Step Designer for the task.
3. Select the step to display the **Step Properties**.
4. Select the custom page from the **Page Layout** menu. The unique identifier for the page is displayed in the **Page Layout** field.

### **Creating routes in a workflow:**

To create routes in a workflow, you draw connectors between steps that define the order in which steps must be completed.

### **Procedure**

To create a route in a workflow:

1. Click the **Add connectors between steps** icon.
2. Click the first step in the route.
3. Drag your mouse to the second step in the route. A connector is added between the two steps. The arrowhead on the connector indicates the direction that the route takes from step to step and the order of the steps.
4. Optional: Select the connector and then specify properties for the connector. If the start step was configured with responses, you can associate a response with the connector. The response is displayed for the connector label in the canvas.

“Splitting and joining a task route”

“Route validation rules for a workflow” on page 61

“Sample routes for a workflow” on page 64

*Splitting and joining a task route:*

Most workflows require branching at various points as the result of a response made by a participant. You use a split to start the branching and a join to bring the work back into a single path at the end of all the true routes.

## About this task

You can select **AND** to assign an AND-split, select **OR** to assign an OR-split, or select **AUTO** to assign the default NONE-split. When you validate the workflow, the system determines the appropriate split type for the AUTO-split based on the steps in the route. That is, the system determines if an AND-split, an OR-split, or a NONE-split is appropriate based on the entire workflow.

**Remember:** For each AND-split step, there must be one AND-join (collector) step. By default, each step has as an OR-join defined.

## Procedure

To create splits and joins for step routes:

1. Create and name the split step.
2. Add one or more responses to the step. For example, assign Yes and No as responses for a step that has two alternate routes.
3. Assign the type of split to the step: OR, AND, or AUTO.
4. Add steps as needed. A branch can contain multiple steps. You must return from an AND-split with a collector step.
5. Create a connector for the route from the split step to the next step.
6. In the **Connector Properties** section, assign a name and responses for the route.
7. Create and name a collector step for the route for each join.
8. Assign the type of join for the collector step.
9. Click **Validate** to validate the steps.
10. Click **Save** to save the step changes for the case type.

*Route validation rules for a workflow:*

The steps and routing in a workflow must be valid before you can deploy the solution. When you are connecting steps, follow the route validation rules to ensure that your workflow is valid without route errors.

In the following diagrams, each step is represented with a circle and each route is represented with an arrow. The arrow indicates the direction of the step processing. The diagrams are simplified versions of how a workflow map looks in the editor.

The following route rules are enforced during workflow validation:

**The workflow starts at the launch step; all steps must be reachable from the launch step.**

Your map must include connectors between steps, and the direction of the route must be consistent.

The following illustration shows an invalid map with steps that cannot be reached:

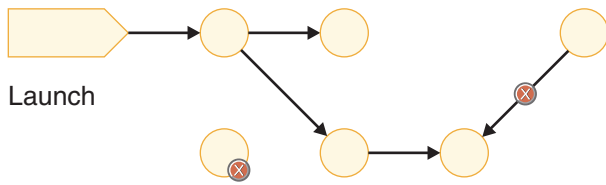


Figure 1. An invalid map with steps that cannot be reached

In the illustration, two steps cannot be reached:

- The step at the lower left is not connected by any route
- The route from the last step on the right goes in the opposite direction of the rest of the route flow, and the step can never be reached.

**Maps must be properly and fully nested with regard to AND-splits and AND-joins, which means that all of the following conditions must be true:**

- For each AND-split step, there must be one AND-join (collector) step. The AND-join step can immediately follow the AND-split step, or there can be one or more steps in between.

The following illustration shows a valid map with all routes from a split meeting at a join step:

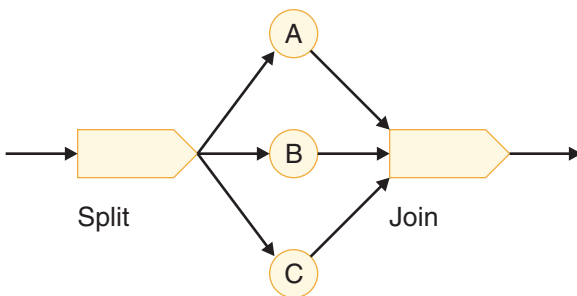


Figure 2. A valid map with all routes from a split meeting at a join step

In the illustration, all three routes from the split meet at the join step.

- All paths from the AND-split step can meet at the AND-join step, or one or more paths can end, that is, stop without going to the AND-join step. A path is defined as a sequence of contiguous routes that can be followed between a set of steps.

The following illustration shows a valid split with one path terminated before join step:

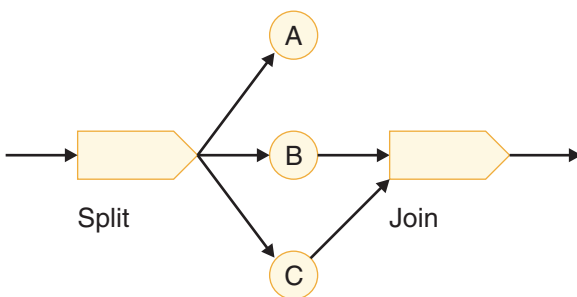


Figure 3. A valid split with one path terminated before the join step

In the illustration, one path from the split step ends at step A, but the paths with steps B and C proceed to the collector step (join). Note that at least one path from the split step must go to the join step.

- A path that passes through an AND-split step cannot return to that step without first passing through the corresponding AND-join step. The following illustration shows an invalid route with one path returning to the split step:

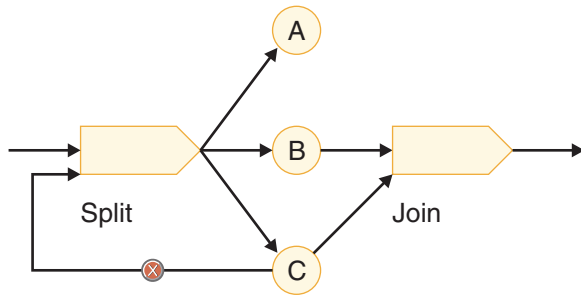


Figure 4. An invalid route with one path returning to the split step

In the illustration, the cycle from step C back to the split step is not valid. Any path from step C must pass through the join step.

- A path that passes through an AND-join step cannot return to that step without first passing through the corresponding AND-split step. The following illustration shows an invalid route that passes only through the join step:

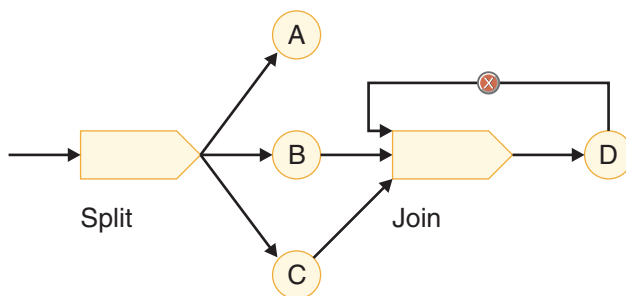


Figure 5. An invalid route from step D that passes only through the join step

In the illustration, the route from step D goes to the join step without passing through the split step, and the cycle from step D to the join step is not valid. To create a valid cycle, the path must first pass through the split step.

- All paths that pass through an AND-join step must first pass through the corresponding AND-split step.

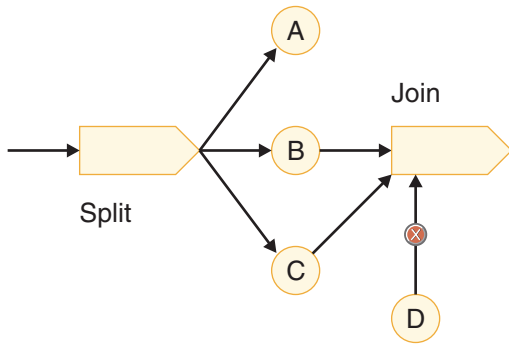


Figure 6. An invalid route connecting directly to join step without passing through split step

In the illustration, the path from step D is not valid because it did not first pass through the split step.

Sample routes for a workflow:

Routes between the steps in a workflow definition specify how work progresses from one step to the next.

Except for the last step on the map, every step has one or more routes leading from it. The following diagrams illustrate simple routes.

**Always true**

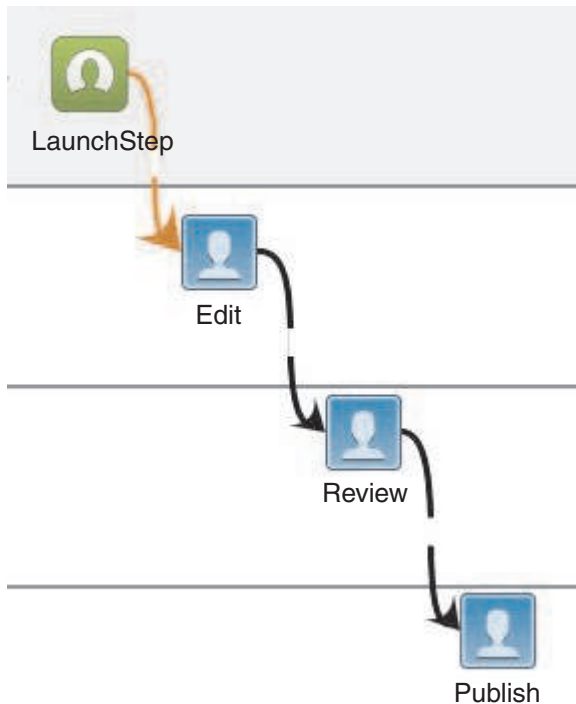


Figure 7. Route with no conditions

Only the most simple workflows proceed linearly from one step to the next. In the illustration in the workflow route, each route is “always true” because there are no conditions to test.

**Tip:** If you define a response for the only route from a step and the condition does not evaluate to true when the step completes, the workflow or the specific branch of the workflow stops.

### Route with an OR condition

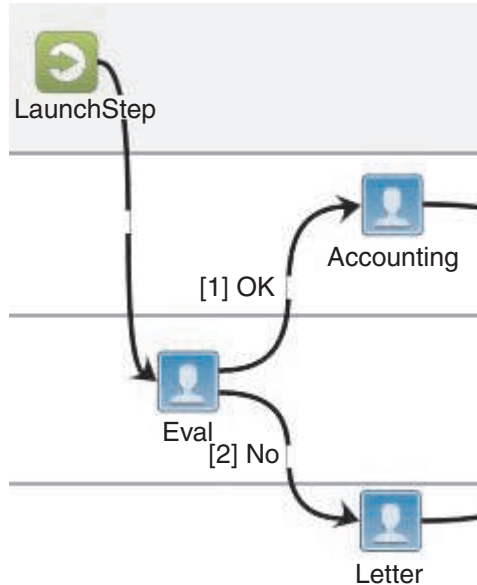


Figure 8. Route with an OR condition

Most workflows require branching at various points as the result of a response made by a participant. In the illustration about the route condition, the route from the launch step is always true. The route from the Eval step depends on the value of a response by the participant at the Eval step. Only one of the routes is taken.

For example, if the Eval step requires the participant to respond by choosing either "OK" or "No," you can define two routes from the step: one route for the OK response and one for the No response. This type of step is called an OR-split.

**Restriction:** Process Designer supports expressions based on property values for a split, but Case Manager Builder does not support expressions for evaluating a step.

### Route with an AND condition

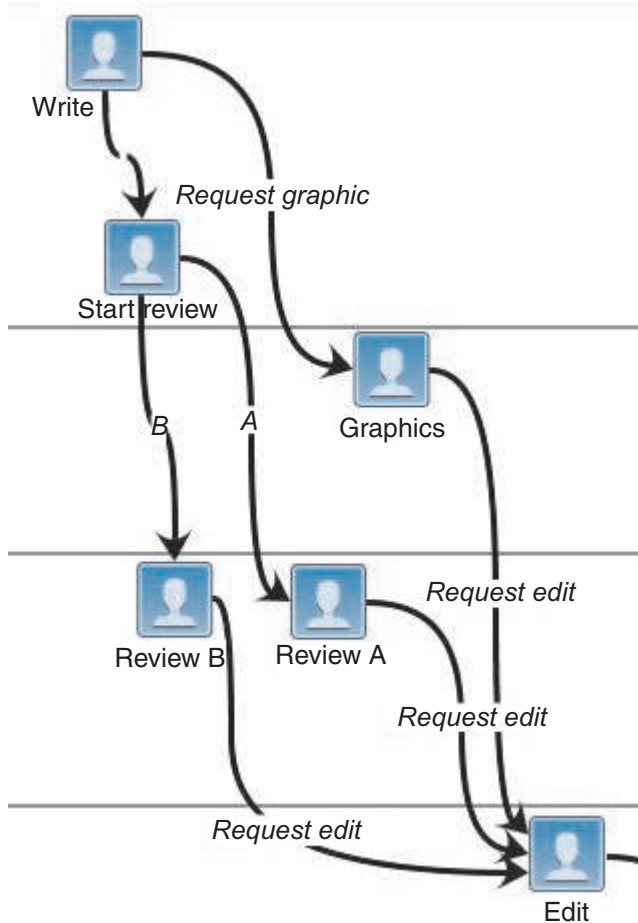


Figure 9. Route with an AND condition

When there are multiple routes from a step and more than one of those routes can evaluate to true, then work can continue down multiple paths simultaneously. The work proceeds along all true routes.

In the illustration about AND conditions, the Write step uses an AND-split. The route to Graphics is always true, and the Start review path and the Request graphics path are both processed. The Start review step is an OR-split that processes either Review A or Review B. The case worker sees that there are two separate work items: one for Graphics and one for either A or B.

To create a valid map, you must define a collector step that brings the work back into a single path at the end of all the true routes. In this illustration, the Edit step is the collector step and is an AND-join.

The processing waits just before the collector step (Edit) until all of the child processes reach this stage.

**Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

**Adding workflows that are not associated with tasks:**

If you want to include a workflow that does not require action by a case worker, you can add a workflow in your case type that is not associated with a task.



## About this task

You can add a workflow to complete a task, or reuse a workflow from a business process management system. You can also add a workflow that does not apply to a specific task in the case type. For example a task workflow might call or create another workflow, or inherit another workflow.

You can include system functions such as `WaitForCondition` in which one workflow calls another workflow that contains information that the first workflow requires for processing steps. The workflow called or created must not be created in the solution collection. You cannot add or edit such a workflow by using Case Manager Builder. Instead, you add or edit this workflow by using Process Designer.

## Procedure

To add a workflow that is not associated with a task:

1. In Process Designer, add a new workflow by clicking **File > Insert > New Workflow**.
2. Specify a name for the workflow on the General page under the **Workflow** tab.
3. Define the new workflow according to your business requirements. For example, add a system step that creates the new workflow in the calling workflow, specify the new workflow as the base workflow in another workflow, or have a task create and then wait for the new workflow.
4. Validate the workflow collection.
5. Insert the workflow into the solution by clicking **File > Solution > Edit**. Then, select the solution definition.
6. Select the case type.

**Important:** Do not select the solution workflow collection.

7. Click **File > Insert > Workflow from Repository**.
8. Do not select to update the base workflow class.
9. Edit the task and workflows according to your business requirements.
10. Validate and save the solution.

## Integrating business processes with IBM Case Manager solutions

If you have existing business processes in FileNet P8 or IBM Business Process Manager, or business rules in IBM Operational Decision Manager, you can integrate them into the solution that you create in IBM Case Manager. You use both Process Designer and Case Manager Builder to integrate the processes into a solution.

“Adding an existing FileNet P8 process as a task” on page 68

“Adding an existing process as a task” on page 70

“Adding a task that runs an IBM Business Process Manager Advanced business process” on page 71

### Related tasks:

 Adding business rules from IBM Operational Decision Manager to a solution

## Adding an existing FileNet P8 process as a task:

You can create a task for a case type from a FileNet P8 process.

### Before you begin

Before you can add an existing FileNet P8 process as a task, you must import one or more processes into a global workflow process collection for the solution by using Process Designer.

### About this task

**Important:** You can change the selected process if you edit the task, but if you select a different process you must map properties again.

### Procedure

To add an existing workflow as a task:

1. From the Tasks page, click **Add Task > Task with Existing FileNet P8 Process**. If **Task with Existing FileNet P8 Process** is not available, then no process is associated with the solution.

**Tip:** If you want to add the process as a subtask inside a container task, open the container task first. Then, click **Add Subtask > Subtask with Existing FileNet P8 Process**.

2. In the General window, specify the name, unique identifier, a description (optional), define how the task starts, and assign the task to a set (optional).
3. In the Preconditions window, define any preconditions that are required before the task can start.
4. In the Select Process window, select the workflow to reuse.
5. In the Map Properties window, map process data fields to solution properties:
  - a. Select a process data field name from the reused workflow.
  - b. Select a solution property name to map to the selected process data field name. Property names must match the data type and number of values (single value or multivalued) of the selected data field. After selecting a data field name, only valid matching properties are displayed in the **Case type property name** menu.
  - c. To add the selected mapping to the property map, click the plus (+) icon.
6. Optional: Click **Finish** or proceed to the final step.
7. Optional: In the Design Comment window, add a design comment that explains the reasons this task was created, or what the imported process does, or how it works with the solution, for example.
8. Click **Finish**.

### Results

You can edit the task in Case Manager Builder, but you can edit the reused process only in Process Designer.

“Importing an existing workflow into a solution” on page 69

“Updating case property values in a reused process” on page 69

“Removing a reused process from a solution” on page 70

*Importing an existing workflow into a solution:*

To reuse an existing IBM Case Foundation workflow as a task, first import the existing workflow into the solution.

### **Before you begin**

Before you can import a workflow, you must have an existing solution with case properties and at least one case type.

### **Procedure**

To import an existing workflow into a solution:

1. In the Manage Solutions page, start Process Designer: Hover over a solution and click **More Actions > Open Process Designer**.
2. In Case Type Selection, select **Solution Workflow Process Collection**.
3. Choose **File > Insert** and select **Local workflow** or **Workflow from repository**.
4. In the directory, browse to the workflow file that you want to insert.


**Restriction:** You must select a IBM Case Foundation workflow XPDL or PEP file. You cannot insert an IBM Case Manager case type XPDL file. You can also edit a workflow or insert a new workflow.


5. Choose to update the base work class of the imported workflow process.
6. Validate the workflow collection.
7. Save and close the solution.

### **Results**

You can now reuse the workflow as a task in Case Manager Builder.

#### **Related information:**

 Cannot access Process Designer from Case Manager Builder with Google Chrome

 Cannot access Process Designer from Case Manager Builder with Google Chrome

*Updating case property values in a reused process:*

You can update case property values as a workflow progresses.

### **About this task**

The global workflow collection contains workflow definitions that do not have a reference to a case type class definition, so these workflow definitions cannot contain expressions that reference case properties.

### **Procedure**

To update case property values:

1. Insert submap steps into the global workflow definition at points where you want to update the case properties.
2. Create submaps in the global workflow definition. Any of the submap steps can call the same submap, or each one can call a different submap, or both.

3. In any of the task workflows that inherit or extend this global workflow definition, override the submaps and add Assignment instructions to update the case or task properties.

*Removing a reused process from a solution:*

You can remove a reused process from a solution by using Process Designer.

### **About this task**

If you added a IBM Case Foundation process to a case type and subsequently want to remove the process, you must remove it from the workflow collection.

### **Procedure**

To remove a reused process from a solution:

1. Open the solution in Process Designer.
2. Optional: If the reused process is the main workflow, set another workflow definition as the main workflow by clicking **Action > Set as Main Workflow**.
3. Select the workflow to delete by clicking **View > Workflows > workflow name** and then selecting **Action > Remove Workflow**.

**Important:** Validation errors occur if you remove a global workflow that is referenced by any task workflows. To avoid the errors, you must modify the task workflows so that they do not reference the removed global workflow.

### **Adding an existing process as a task:**

You can create a task for a case type from an existing process that was defined in IBM Business Process Manager.

### **Before you begin**

Before you can add an IBM Business Process Manager process as a task, a process application of the same name must be associated with the solution, and you must add a case type to the solution.

### **Procedure**

To add an IBM Business Process Manager process as a task:

1. Open the solution that you want to add the task to in Case Manager Builder.
2. On the Tasks page, click **Add Task > Task with Existing Process**. If **Task with Existing Process** is not available, then no process is associated with the solution.

**Tip:** If you want to add the IBM BPM process as a subtask inside a container task, open the container task first. Then, click **Add Subtask > Subtask with Existing Process**.

3. On the General page, specify the name, unique identifier, and a description (optional), define how the task starts, and assign the task to a set.
4. On the Preconditions page, define any preconditions that are required before the task can start.
5. On the Select Process page, select the process to reuse. The available processes depend on the process application that is associated with the solution.

To add a process, click **Open Web Process Designer** to open IBM BPM Process Designer. Add the process, create a snapshot, and make that snapshot the default. To view any processes that were added after you opened the Add Task dialog box, click **Refresh**.

To edit a process, click **Open Web Process Designer**, edit the process, and update the default snapshot. The updated snapshot is saved in the solution definition file when you open the solution for editing.

6. On the Map Properties page, map process data fields to solution properties:
  - a. Select a process data field name from the reused process.
  - b. Select a solution property name to map to the selected process data field name. Properties must match the data type and number of values (single value or multivalued) of the selected data fields. For example, string properties only match string data fields, and single value properties only match single value data fields. After you select a data field name, only valid matching properties are displayed in the **Property name** menu.
  - c. To add the selected mapping to the property map, click the plus (+) icon.
7. Optional: In the Design Comments page, add a design comment. For example, the comment might explain the reasons this task was created, what the imported process does, or how it works with the solution.
8. Click **Finish**.

## Results

You can view the associated process by clicking the **Open Web Process Designer** icon for the task. This icon opens the process in IBM BPM Process Designer.

**Important:** You can change the selected process if you edit the task later, but if you select a different process you must map properties again.

## Adding a task that runs an IBM Business Process Manager Advanced business process:

You can create a task in Case Manager Builder that runs as a business process on IBM Business Process Manager Advanced.

### About this task

The business process is implemented as an external web service. The IBM Case Manager task uses the web service to integrate with the business process running on Business Process Manager Advanced. The business process performs the work that drives the outcome of the task.

## Procedure

To add a task that runs a business process:

1. Ensure that Business Process Manager Advanced is configured and is running.
2. In Case Manager Builder, create a case management solution and add the task that runs the business process.

Add the task to the appropriate case type in the same way that you add other tasks, except do not use the Step Designer to create steps for the task. Instead, create these steps by using Integration Designer.

3. In Integration Designer, use the external service wizard to create a web service to implement the case management task.

In addition to creating the web service, Integration Designer updates the partner link reference in the solution XPDL file. The partner link reference identifies the endpoints to the web service, which links the task and web service.

4. Deploy the module that contains the web service to Business Process Manager Advanced in the development environment.
5. In Case Manager Builder, deploy and test your solution in the development environment.
6. Validate the task:
  - a. Add a case of the case type that contains your task.
  - b. Start the task.
  - c. In Process Task Manager, expand the **Component Managers** node to ensure that the component manager is started for the task.
7. Continue to edit your solution and business process until your business requirements are met.

Use Case Manager Builder to update the case type and to test the changes.

Use Integration Designer to update the business process. As you modify input and output parameters, use the Edit Binding function to update the web service with the changes.
8. Deploy the web service module to Business Process Manager Advanced in the target environment, for example, to a test environment or a production environment.
9. Deploy the solution to the target environment:
  - a. Use FileNet Deployment Manager to create the solution definition for the target environment.

You must have a different FileNet P8 domain for each Business Process Manager Advanced runtime server.

You must update the PEPartnerLink attribute in the service data map with the endpoint for the web service module on the production server.
  - b. Use the IBM Case Manager administration client to deploy the solution to the target environment.
10. Use the administrative console of Business Process Manager Advanced to make changes to the module properties.

### What to do next

By default, the IBM FileNet P8 Component Manager service user credentials are used for authentication with a web service on Business Process Manager Advanced. If the web service requires different credentials, you must configure the credentials in Process Task Manager.

If you have long-running business processes (macroflows) and you want to provide Secure Socket Layer (SSL) support, you must do these tasks:

- Configure SSL on the Application Engine server on which the Component Manager is running.
- In Process Task Manager, configure the listener URL for the Component Manager with the HTTPS endpoint that is used to start the web service. This endpoint is sent in the WS-Addressing ReplyHeader element of the request when a task initially invokes the business process.

**Important:** SSL support in the FileNet P8 environment is a system wide configuration that governs all web service traffic into Process Engine. Therefore, you must consider the ramifications on your overall system before you enable SSL.

## Designing views for to-do tasks

You can specify which properties to display for a to-do task by designing its view.

### About this task

Defining a view for a to-do task is optional. You can use the default view instead. The default view provides a list of all task properties that are associated with the to-do task.

In some cases, you might want to customize the view for a to-do task. For example, you might want to display only a subset of the associated task properties, change the order in which the properties are displayed, or include case properties. You can also customize the layout of the view.

As you design a view, consider the space that is available in the user interface. If the widget is too wide or too high, you might see scroll bars that make editing the properties awkward for the user. Use the different layout containers such as the tabbed layout container or the titled layout container to organize the content of the view efficiently.

**Attention:** If you set a custom size for the Properties View Designer window, minimizing or maximizing the window causes Properties View Designer to display incorrectly.

### Procedure

To customize the view for a to-do task:

1. On the Tasks page for the case type, click the **Open To-do View Designer** icon for the to-do task.
2. Drag one or more containers onto the canvas. Each container provides a different layout for the properties that it contains. For the layout containers, you can add other containers and properties to a container.

**Tip:** To make working with the containers easier, the design view displays extra space around the containers. Click the **Hide Extra Padding** icon see an approximation of how the view will look in Case Manager Client.

3. Select a container and configure the settings for that container. The tabbed layout container and the multiple column container consist of a set of layout containers that are arranged in either tabbed pages or columns. You can configure settings for the entire container and for the individual layout containers.
4. Drag the properties into the appropriate containers. The **Properties** palette includes all the properties that you added to the task and the case type.

#### Tips:

- Multiple selections are supported by using the Shift and Ctrl keys.
- Properties are added to the new container in the order in which they were selected.
- Properties can be dragged between the containers in the view.
- Properties cannot be dragged into the child containers of a view.

- After you start a drag operation, press the Ctrl key to copy the objects instead of moving the objects. Press the ESC key to cancel a drag operation.
  - The graphic changes to indicate whether an object or group of objects can be dropped into the container.
5. Select a property and configure the settings for that property.

When you add a property to a container, it initially uses the default control for editing the property. For example, a Boolean property is displayed as a check box by default. You can change the type of control that is used for a property. For example, you might change the control so that a Boolean property displays as radio buttons or a drop-down list. You can also select **Static text** to display the property value in read-only mode and configure the appearance and behavior of some controls.

**Important:** You can set a default value for each property in the view. However, this value is applied only if the default value in the property is null and is not specified by another mechanism such as a JavaScript call or an external data service.

6. Save the view and close the designer.

### What to do next

To display the to-do task view in your solution, add a To-do List widget to a page. Then, edit the To-Do List widget settings to select the to-do tasks to display.

#### Related concepts:

“Property editor settings” on page 28

“To-Do List widget” on page 124

#### Related tasks:

“Adding external properties to the properties view” on page 27

#### Related reference:

“Containers” on page 24

“Properties” on page 29

---

## Multiple user editing of solutions

Multiple users can edit a single solution at the same time. Opening an asset for editing locks the asset so that only the user who is editing can change it. In addition, editing certain assets locks all associated assets.

Some solution assets are associated with specific configuration files. If any one of these assets is being edited, the other assets that are associated with the same file are unavailable for editing. When multiple users are editing a solution, the behavior for the various solution assets is as follows:

*Table 8. Availability of solution assets*

Asset	Availability
Pages	A page can be modified by only one user at a time. If a page is being edited, it is locked until the page is saved and committed. Users can create or edit other pages while a page is being edited.
Business rules	A business rule can be modified by only one user at a time. If a business rule is being edited, it is locked until the business rule is saved and committed. Users can create or edit other business rules while a business rule is being edited.



Table 8. Availability of solution assets (continued)

Asset	Availability
Views	A view can be modified by only one user at a time. If a view is being edited, it is locked until the view is saved and committed. Users can create or edit other views while a view is being edited.
Process Engine configuration file assets: in-baskets or roles	If any asset that is associated with the Process Engine configuration file is being edited, all similar assets are locked. Only one user can edit assets from Process Engine at one time.
Solution definition file (SDF) assets: properties, document classes, case types, case folders, case summary view, case search view, choice lists, solution descriptions, or solution icons	If any asset that is associated with the SDF is being edited, all similar assets are locked. Only one user can edit any SDF assets at one time. When another user is editing an SDF asset, some SDF assets might be available for viewing only.
Solution workflow definition file assets: tasks	If any tasks that are associated with the solution workflow definition file are being edited, all similar assets are locked. Only one user can edit a task at one time.

When you save changes to an asset, those changes are saved as drafts until you commit the changes. If you deploy the solution before the changes are committed, the draft changes are not included in the deployment. When lists of available items are displayed in Case Manager Builder, such as attachments and predefined workgroups, the lists include committed items only.

**Tips:**

- If you want the list of available items to include an item that you created in another task but did not yet commit, first open the task with which the item is associated. For example, you previously added an attachment to task A but did not yet commit the changes. If you want to add that attachment to task B, ensure that you first open task A before you open task B.
- Before you commit your solution, you can click **Validate** on the solution home page to ensure that none of the following assets were deleted by another user while you were editing the solution:
  - Properties, roles, or step pages that are used in a workflow
  - Properties that are used in an in-basket
  - Tasks or case pages in a case type

For assets that are associated with a configuration file, the file is locked when an asset is created in the file. For example, if you are creating a new property or editing an existing one, the file is not locked until you click **OK**.

**Important:** For all assets other than case types and tasks, the asset is unlocked after you view it or discard any changes that you made. However, if you open a case type or task for editing and discard your changes instead of saving them, the asset remains locked by you. You must commit your changes to the solution even if you discarded the changes to the case type or task.

---

## Designing the case management client application

The client application consists of a set of pages that are displayed in Case Manager Client for your solution. Each page contains the widgets required to perform a specific action or task. IBM Case Manager provides a set of default pages that you can customize to meet the requirements of specific roles, case types, or work items in your solution. Alternatively, you can create custom pages to use in place of the default pages.

“Creating a custom page”

“Widget, action, and page events and wiring” on page 128

## Creating a custom page

IBM Case Manager includes a set of pages that you can use in your solution. You can customize these pages to meet the requirements of your solution. You can use these pages as templates to create additional pages to provide customized views for different roles, case types, and tasks.

### About this task

By creating custom pages, you can tailor the Case Manager Client user interface to meet specific requirements. For example, you might want two versions of the Case Details page, one for managers and one for case workers. In the version that is intended for managers, you include the Timeline Visualizer widget to display the extended case history. In the version that is intended for case workers, you do not include the Timeline Visualizer widget.

You create and test new pages in your development environment. When you deploy your solution to a production environment, the new pages are automatically copied to that environment

**Tip:** For best results when you designing pages, use a monitor that is the same width and resolution as the monitors that your users will be using at run time.

### Procedure

To create a page:

1. Open the solution in Case Manager Builder.
2. On the **Pages** tab, add a page of one of the following page types:

#### **Solution**

Creates a blank page.

#### **Case Details**

Creates a page that is based on the IBM Case Manager Case Details page

#### **Add Case**

Creates a page that is based on the IBM Case Manager Add Case page

#### **Split Case**

Creates a page that is based on the IBM Case Manager Split Case page

#### **Add Task**

Creates a page that is based on the IBM Case Manager Add Task page

#### **Work Details**

Creates a page that is based on the IBM Case Manager Work Details page

#### **Custom Task Details**

Creates a page that is based on the IBM Case Manager Custom Task Details page

To create a page that is based on one of the other IBM Case Manager pages or on a custom page, click the **Copy** icon for that page.

3. Click the page name to design and configure the page layout.

4. In Page Designer, design the page layout:
  - a. On the toolbar, click the **Page Options** icon to open the Page Options window, where you can select and configure the page layout. Click **OK**.  
You can choose a layout that has a set configuration of rows, columns, or both. Alternatively, you can choose the **Free Form** layout. With this layout, you use the **Column** and **Tab** containers to configure the page exactly as you need.
  - b. If you selected the **Free Form** layout, drag containers onto the page.
  - c. Drag widgets into the appropriate containers on the page.

**Restriction:** The solution pages do not monitor for unsaved changes in the widgets that are on the page. To avoid potential loss of user input, do not put the Properties widget, Form widget, or Viewer widget on these pages.

- d. Edit the settings for the widgets.
  - e. Edit the wiring for the page and the widgets, if needed.
5. Associate the page with the role, work item, or case type:

Component	Action
Case type	<p>Specify the new page layout as a default page for a case type:</p> <ol style="list-style-type: none"> <li>1. On the <b>Case Types</b> tab, open the case type.</li> <li>2. Select the new page as the default layout for the appropriate page type:               <ul style="list-style-type: none"> <li>• Work Details for Custom Task page</li> <li>• Add Case page</li> <li>• Split Case page</li> <li>• Case Details page</li> </ul> </li> </ol> <p>To set the new page as the default Case Details or Case Details Form page for a specific role, click <b>Add Role</b> and select the page.</p>
Role	<p>Specify the new page layout as a Solution page for a role:</p> <ol style="list-style-type: none"> <li>1. On the <b>Roles</b> tab, open the role.</li> <li>2. On the <b>Pages</b> tab for the role, click <b>Assign Page</b> and select the page.</li> </ol>
Work item	<p>Specify the new page layout as the default Work Details page for a step in a task:</p> <ol style="list-style-type: none"> <li>1. On the <b>Case Types</b> tab, open the case type.</li> <li>2. On the <b>Tasks</b> page, open the task that contains the step.</li> <li>3. Open the task in Step Designer and select the step.</li> <li>4. In the <b>Step Properties</b> area, select the page from the <b>Page Layout</b> list.</li> </ol>

“Pages” on page 78

“Widgets” on page 91

**Related concepts:**

“Pages”

## Pages

A *page* contains the widgets that are required to complete an activity. For example, the widgets that you use to create a loan request are grouped on a page.

A page consists of a layout, widget configurations, and the flow of events between widgets. The flow of events is achieved either by widgets that broadcast events or by widgets that are wired to other widgets on the page. You might configure your application so that each case task has its own page. For example, you might require that a loan officer must create and submit loan requests and notify customers when a decision is reached. So you add a page to create and submit loan requests. You then create a separate page with work items to contact customers with the status of their loan requests. In addition, you might configure the application so that different roles use different pages.

An HTML template defines the layout and widget configuration for a page. A JavaScript file contains the Dojo classes and IBM Case Manager JavaScript classes that handle the processing that is required for the page. A JSON file contains the definitions for wired events and any disabled broadcast events.

“Solution pages”

“Case Details pages” on page 80

“Add Case pages” on page 82

“Split Case pages” on page 84

“Add Task pages” on page 85

“Work Details pages” on page 87

“Custom Task Details pages” on page 90

### Related concepts:

“Case Details pages” on page 80

“Add Task pages” on page 85

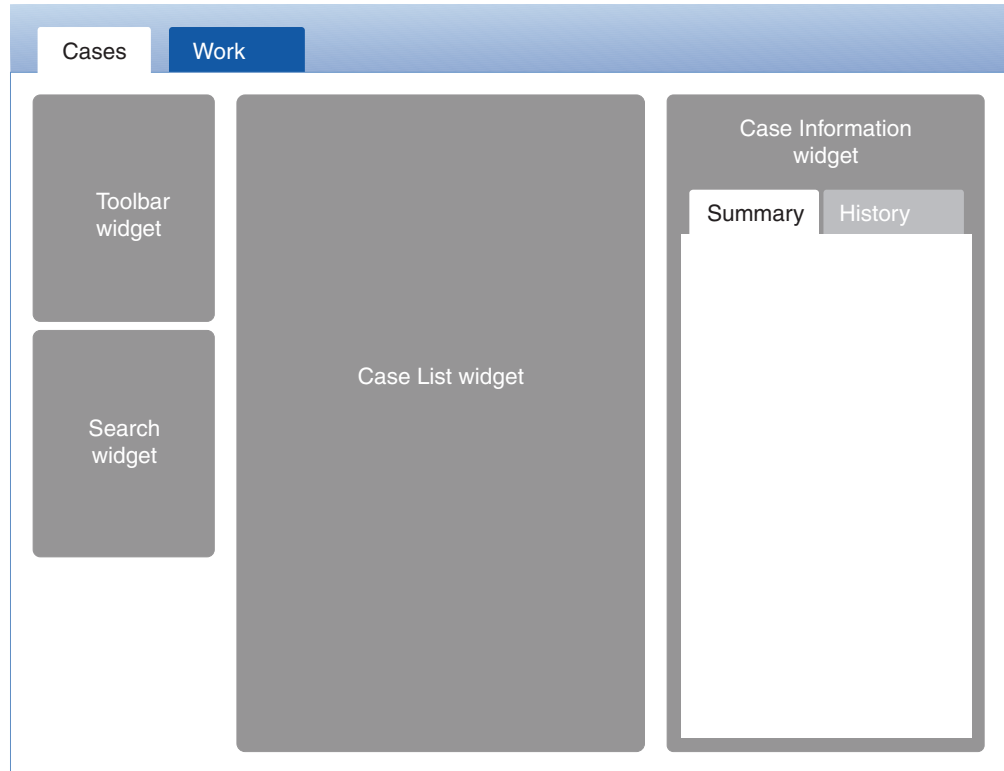
### Solution pages:

Solution pages provide case workers with access to cases and work items that are in a solution. The solution pages include the Work page, with the In-baskets and Toolbar widgets, and the Cases page, with the Case Search-related widgets.

IBM Case Manager provides two default solution pages, the Cases page and the Work page.

### Cases page

By default, the Cases page contains the widgets that are laid out as shown in the illustration.



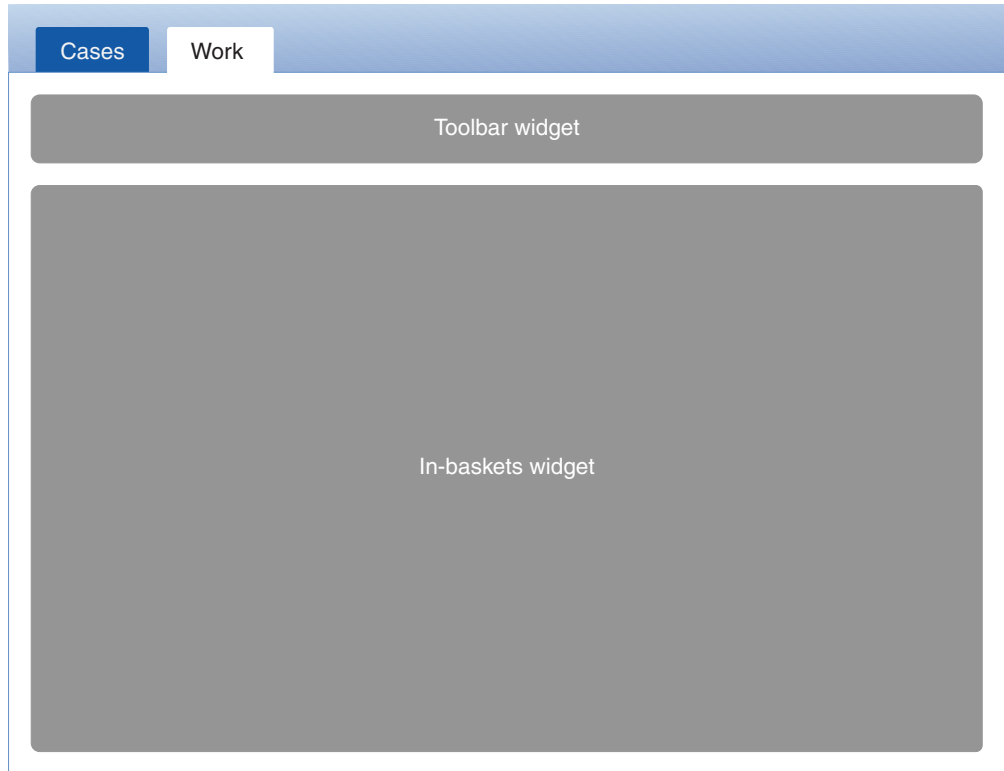
When a user first opens the Cases page, the Case Information widget is hidden. The Cases page contains two hidden Script Adapter widgets that are used to display the Case Information widget when a user selects a case.

By default, when included on the Cases page:

- The Toolbar widget includes the **Add Case** button.
- The Case Information widget includes the following tabbed views: Summary and History.

### **Work page**

By default, the Work page contains the widgets that are laid out as shown in the illustration.



By default, when included on the Work page, the Toolbar widget includes the following buttons: **Manage Roles** and **Add Case**.

**Related concepts:**

“Toolbar widget” on page 124

“In-baskets widget” on page 112

“Search widget” on page 122

“Case List widget” on page 101

“Case Information widget” on page 97

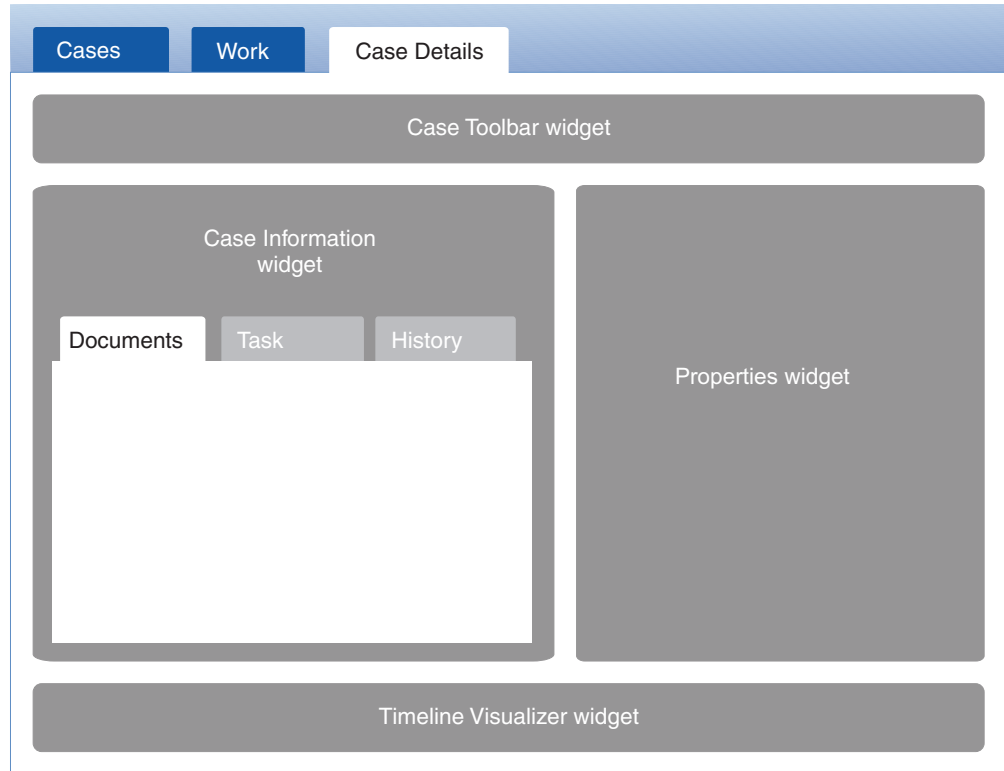
**Case Details pages:**

The Case Details pages include pages that case workers use to interact with specific cases. IBM Case Manager provides two default Case Details pages, the Case Details page and the Case Details Form page.

**Case Details page**

The Case Details page is used for viewing or updating case properties.

By default, the Case Details page contains the widgets that are laid out as shown in the illustration.



By default, when included on the Case Details page:

- The Case Toolbar widget includes the following buttons: **Add Custom Task**, **Add Task**, **Close**, **Comments**, **Create Package**, **Save**, and **Split Case**. The **Add Custom Task** button opens a menu that includes two options: **New** and **Copy Existing Custom Task**.

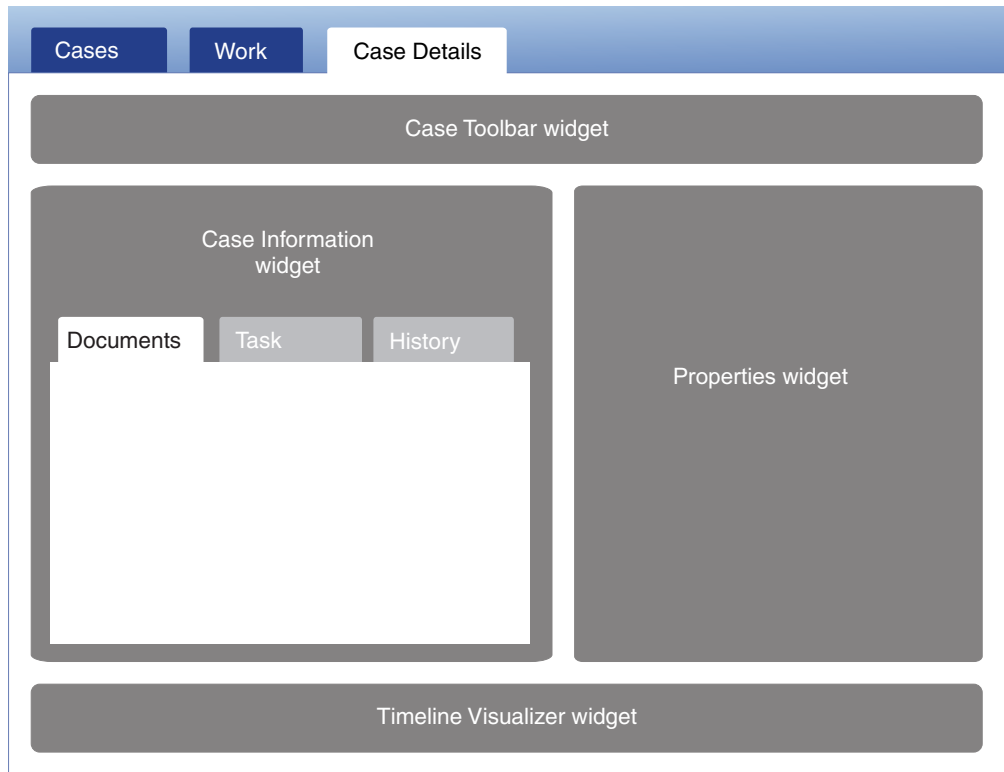
**Restriction:** The **Create Package** button is included by default only for solutions that are created with version 5.2.1 Fix Pack 4 or later.

- The Case Information widget displays the following tabbed views: **Documents**, **History**, and **Tasks**.

### Case Details Form page

The Case Details Form page is used for viewing or updating case properties in the Form widget instead of the Properties widget.

By default, the Case Details Form page contains the widgets that are laid out as shown in the illustration.



By default, when included on the Case Details Form page:

- The Case Toolbar widget includes the following buttons: **Add Custom Task**, **Add Task**, **Close**, **Comments**, **Create Package Save**, and **Split Case**. The **Add Custom Task** button opens a menu that includes two options: **New** and **Copy Existing Custom Task**.
- The Case Information widget displays the following tabbed views: **Documents**, **History**, and **Tasks**.

**Related concepts:**

“Case Toolbar widget” on page 103

“Case Information widget” on page 97

“Viewer widget” on page 125

“Properties widget” on page 117

“Select Case Documents widget” on page 123

**Add Case pages:**

The Add Case pages includes pages that case workers use to create cases of specific case types. IBM Case Manager provides two default Add Case pages, the Add Case page and the Add Case Form page.

**Add Case page**

The Add Case page is used for creating cases.

By default, the Add Case page contains the widgets that are laid out as shown in the following illustration.



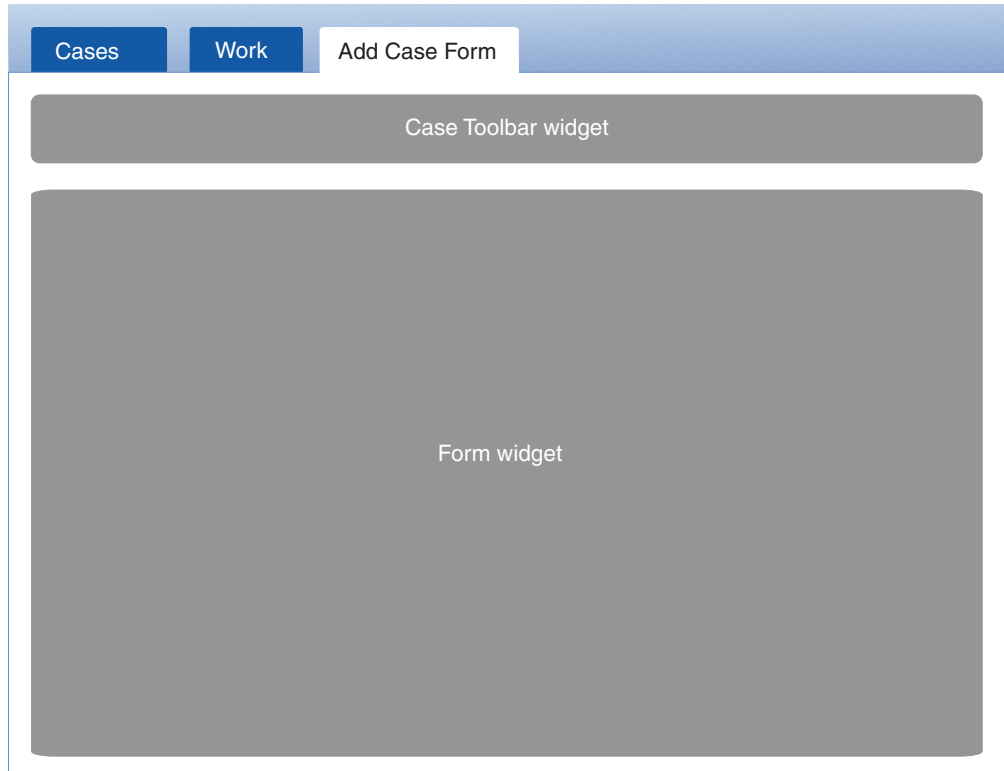


By default, when included on the Add Case page, the Case Toolbar widget includes the following buttons: **Add** and **Cancel**.

### **Add Case Form page**

The Add Case Form page is used for creating cases. On this page, the Properties widget is replaced by the Form widget where case properties are displayed.

By default, the Add Case Form page contains the widgets that are laid out as shown in the following illustration.



By default, when included on the Add Case Form page, the Case Toolbar widget includes the following buttons: **Add** and **Cancel**.

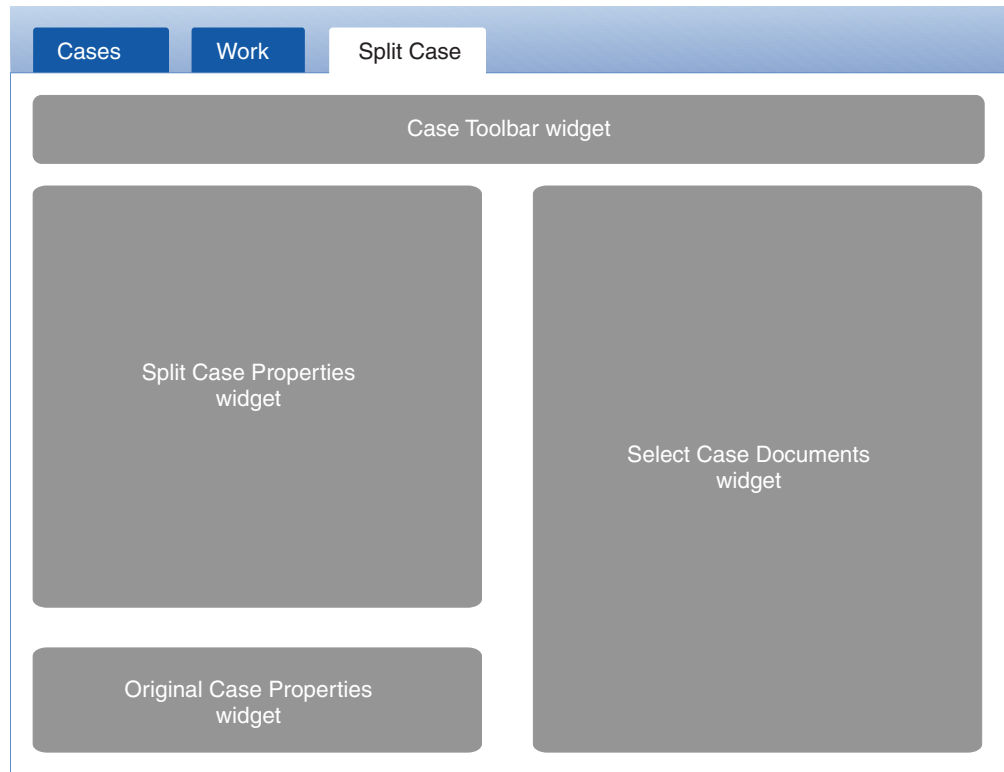
### **Split Case pages:**

The Split Case pages include pages that case workers use to create new cases that are based on existing cases. IBM Case Manager provides one default Split Case page.

### **Split Case page**

The Split Case page is used for creating new cases by reusing data from existing cases.

By default, the Split Case page contains the widgets arranged as shown in the following illustration.



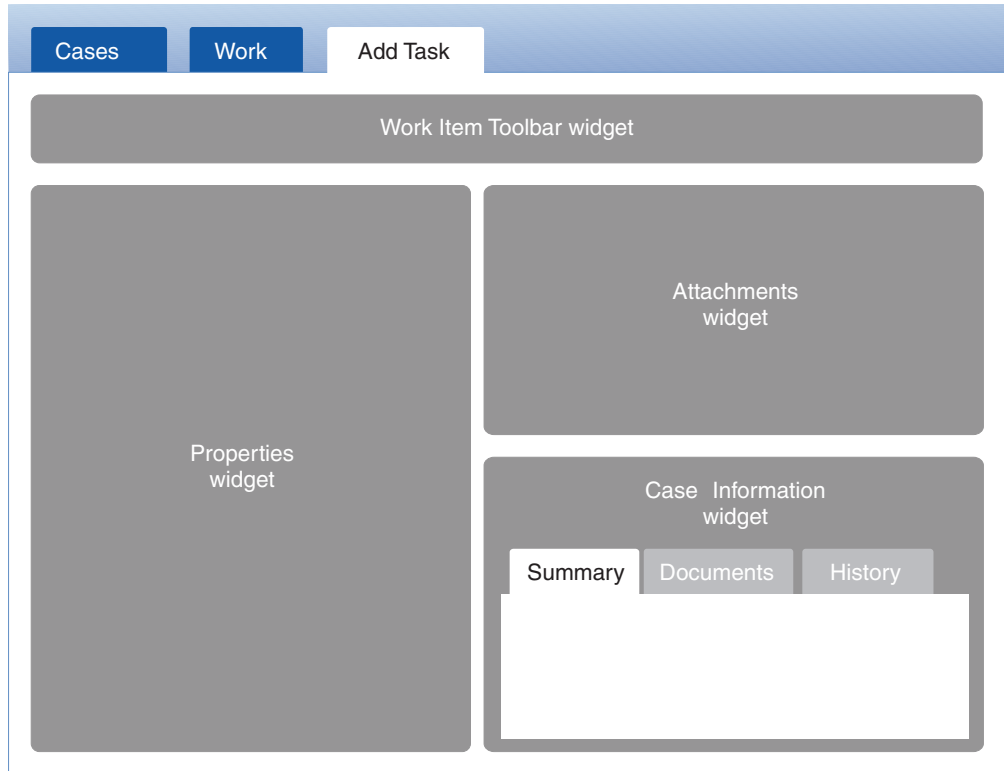
By default, when included on the Split Case page, the Case Toolbar widget includes the following buttons: **Add** and **Cancel**.

#### **Add Task pages:**

The Add Task pages include the step pages that display when a case worker adds a discretionary task to a case. IBM Case Manager provides two default Add Task pages, the Add Task page and the Add Task Form page.

#### **Add Task page**

By default, the Add Task page contains the widgets that are laid out as shown in the illustration.

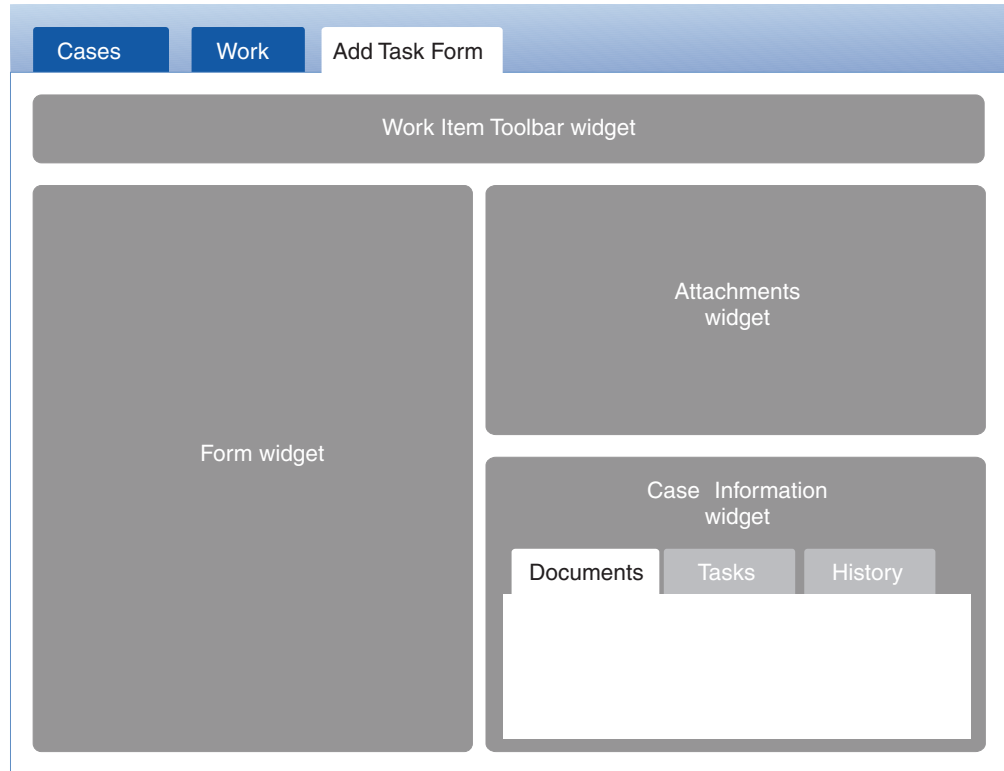


By default, when included on the Add Task page:

- The Work Item Toolbar widget includes the following buttons: **Add** and **Cancel**.
- The Case Information widget displays the following tabbed views: **Documents**, **History**, and **Tasks**.

#### **Add Task Form page**

By default, the Add Task Form page contains the widgets that are laid out as shown in the illustration.



By default, when included on the Add Task Form page:

- The Work Item Toolbar widget includes the following buttons: **Add** and **Cancel**.
- The Case Information widget displays the following tabbed views: **Documents**, **History**, and **Tasks**.

**Related concepts:**

“Work Item Toolbar widget” on page 126

“Properties widget” on page 117

“Viewer widget” on page 125

“Attachments widget” on page 93

“Case Information widget” on page 97

“Form widget” on page 108

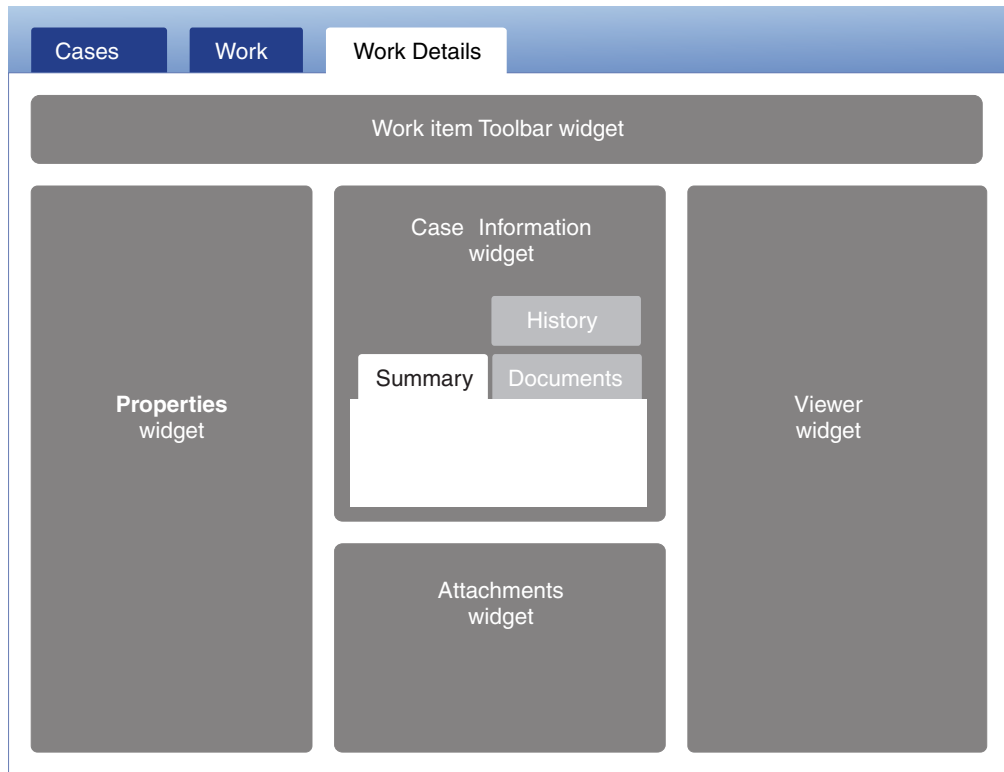
**Work Details pages:**

The Work Details pages includes step pages that display when a case worker opens a work item. Case workers use the step pages to complete the work items that are assigned to them.

IBM Case Manager provides three default Work Details pages: the Work Details page (for working on a work item), the Work Details Form page (for working on a work item by using a form), and the Form Attachment Work Details page (for working on a form-based work item).

**Work Details page**

By default, the Work Details page contains the widgets that are laid out as shown in the illustration.

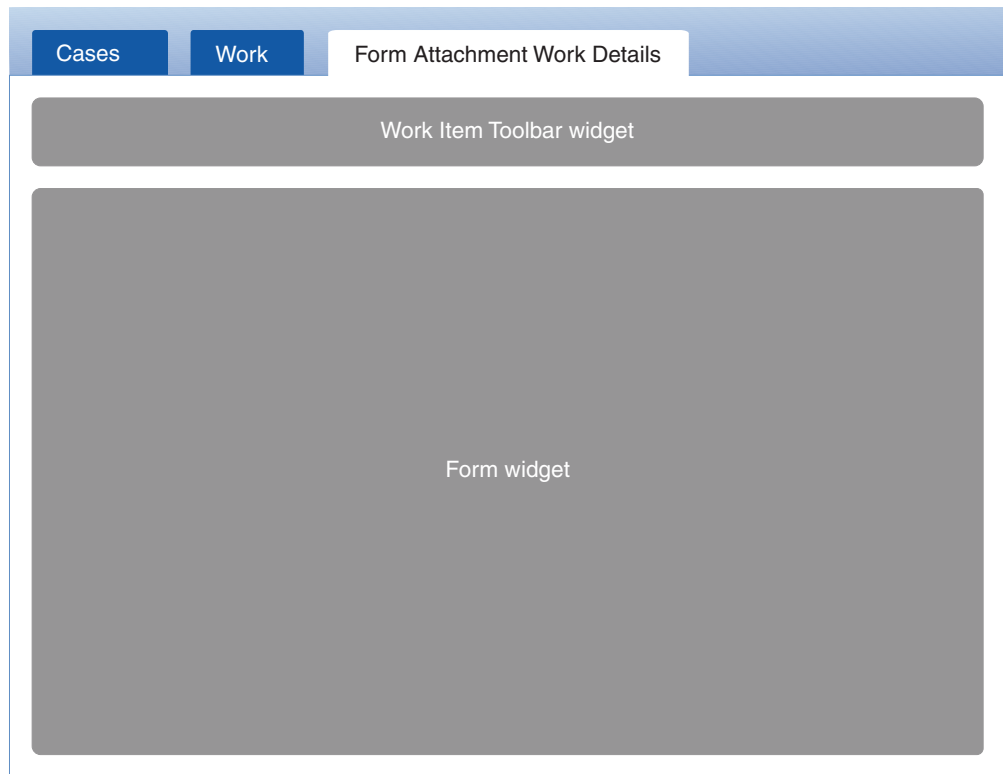


By default, when included on the Work Details page:

- The Work Item Toolbar widget includes the following buttons: **Close**, **Comments**, **Complete**, and **Save**.
- The Case Information widget displays the following tabbed views: **Documents**, **Tasks**, and **History**.

#### **Form Attachment Work Details page**

By default, the Form Attachment Work Details page contains the widgets that are laid out as shown in the illustration.

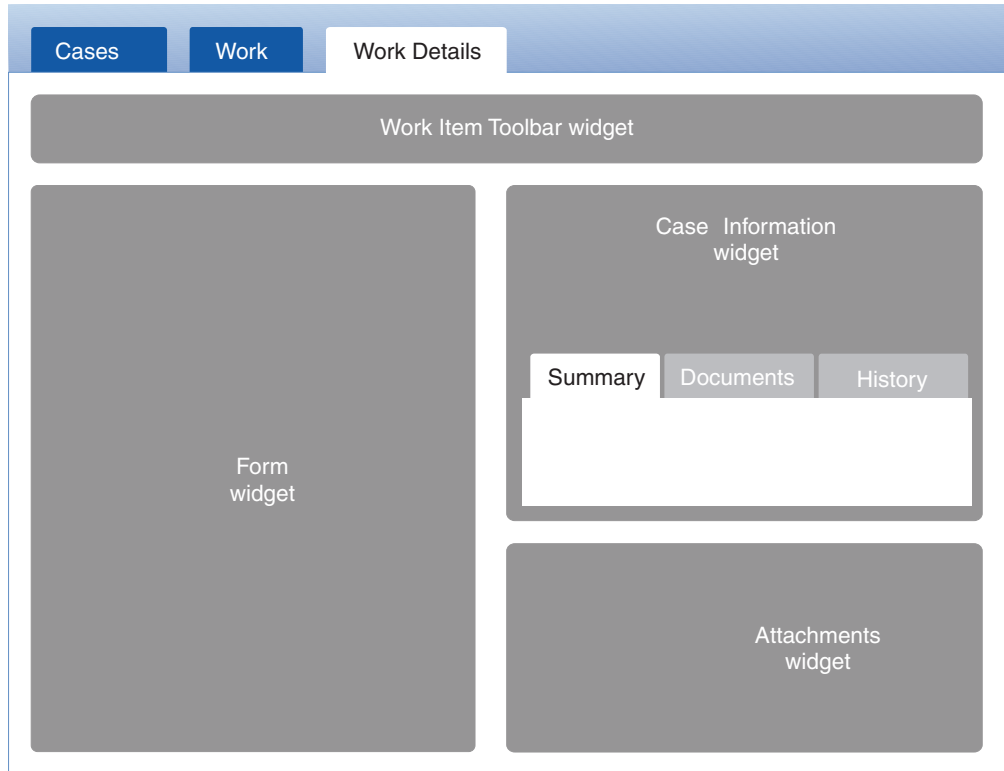


The information that a case worker enters in the Form widget on this page is saved in a form document. The form document is then attached to the work item.

By default, when included on the Form Attachment Work Details page the Work Item Toolbar widget includes the following buttons: **Close**, **Comments**, **Complete**, and **Save**.

### **Work Details Form page**

By default, the Work Details Form page for forms contains the widgets that are laid out as shown in the illustration.



The information that a case worker enters in the Form widget on this page is saved as property values for the case.

By default, when included on the Work Details page:

- The Work Item Toolbar widget includes the following buttons: **Close**, **Comments**, **Complete**, and **Save**.
- The Case Information widget displays the following tabbed views: **Documents**, **Tasks**, and **History**.

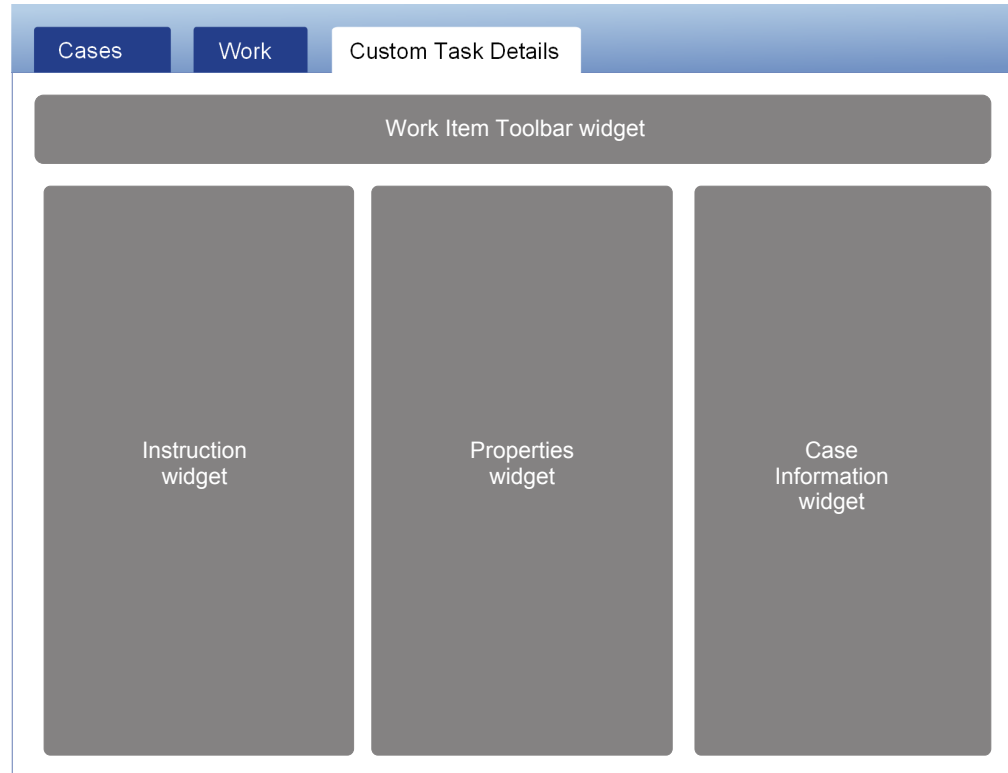
#### **Custom Task Details pages:**

The Custom Task Details pages include the step page that displays when a case worker opens a work item that was defined for a custom task. IBM Case Manager provides one default Custom Task Details page.

#### **Custom Task Details page**

By default, the Custom Task page contains the widgets that are laid out as shown in the illustration.





By default, when included on the Custom Task Details page:

- The Work Item Toolbar widget includes the following buttons: **Close**, **Comments**, **Complete**, and **Save**.
- The Case Information widget displays the following tabbed views: **Documents**, **Tasks**, and **History**.

## Widgets

Widgets are portable, reusable components that are used in pages to manage content and process work items.

The default pages in Case Manager Client include widgets that are already wired together. You can wire other widgets together, including widgets from other vendors, to create a customized application.

For example, if you want to create an application for a bank loan department, you can configure the application so that bank clerks can initiate loan applications, and managers can approve or deny those applications. To create this application, you create a Solution space for the bank by copying the default Solution space. The default Solution space contains both Work page and Cases page. The Work page includes the Toolbar and In-baskets widgets. The Cases page includes the Toolbar, Search, Case List, and Case Information widgets.

In our example bank loan application, you might use the widgets to provide the following functions:

### In-baskets widget

Use the In-baskets widget to allow bank clerks and managers to view the loan applications that for the Clerk or Manager roles. Most of the work is completed using the In-baskets widget.

**Toolbar widget**

From the Toolbar widget clerks or managers can start a new loan application. Managers can add new employees to the case worker role from this widget. With the Toolbar widget, you can allow switching between different roles if the clerk, manager, or other bank worker is a member of different roles.

**Case Information widget**

The Case Information widget displays information about the loan application case in a tabbed format. In the Summary view, clerks can view the associated information for work items. In the Documents view, clerks can do various document-related tasks including: add, view, remove, add comments to a document, and check in or out any document associated with the case.

**Search widget**

From the Search widget bank clerks and managers can conduct various searches for loan applications.

**Case List**

The Case List widget displays the list of loan applications.

There are two additional spaces, Case Pages space and Step Pages space, that also include various pre-wired widgets.

After creating space and pages, you can modify the location of widgets on the pages, add widgets onto pages, and configure the widgets.

- “Attachments widget” on page 93
- “Calendar widget” on page 96
- “Case Information widget” on page 97
- “Case List widget” on page 101
- “Case Stages widget” on page 103
- “Case Toolbar widget” on page 103
- “Chart widget” on page 105
- “Content List widget” on page 105
- “Form widget” on page 108
- “In-baskets widget” on page 112
- “Instruction widget” on page 114
- “Markup widget” on page 115
- “Original Case Properties widget” on page 116
- “Process History widget” on page 116
- “Properties widget” on page 117
- “Script Adapter widget” on page 120
- “Search widget” on page 122
- “Select Case Documents widget” on page 123
- “Split Case Properties widget” on page 123
- “Timeline Visualizer widget” on page 123
- “To-Do List widget” on page 124
- “Toolbar widget” on page 124
- “Viewer widget” on page 125
- “Website Viewer widget” on page 126

“Work Item Toolbar widget” on page 126

**Related concepts:**

“Solution pages” on page 78

“Case Details pages” on page 80

“Add Task pages” on page 85

**Attachments widget:**

You use the Attachments widget to display a list of the documents that are attached to a work item. The documents can be stored either on the Content Platform Engine server or in the IBM Content Manager repository, depending on your configuration. In addition, you can configure a case type so that the Attachments widget can display documents from external repositories.

For FileNet P8 documents, the Attachments widget lists attachments by the document title that is assigned when the attachment was added to the widget. This title is automatically updated when the document title is updated in the repository.

In addition to viewing attachments, users use the Attachments widget to:

- Upload local files to a repository
- Attach documents by browsing and selecting them from a repository
- Attach documents by dragging them from your file system to the Attachment widget
- Check out, check in, cancel checkout, and view properties of documents; download attachments; remove attachments
- Add documents from external repositories as work item attachments

**Restriction:** Users can select any item to add to the Attachments widget, including system files such as workflows and entry templates. However, the check out, check in, cancel checkout, and view properties actions are supported only for document classes. Using these operations for other file types can produce unexpected results.

The fields that are displayed on the Attachments widget and the actions that you can take are determined by the following permissions, which are assigned to the fields when the workflow is created or modified in the FileNet Process Designer:

**Read** The Attachments widget displays the attachment name and the name of the assigned target document. You can view the target document, but you cannot change it.

**Write** The Attachments widget displays the attachment name, but not the name of the assigned target document. Optionally, you can change to a different target document for the attachment.

**Read/Write**

The Attachments widget displays the attachment name and the name of the assigned target document. You can view the target document and, optionally, you can change to a different target document for the attachment.

You can edit the settings for the Attachments widget to:

- Specify whether attachment content will be displayed in the Viewer widget or in a separate browser window

- Specify whether to display a list of all document classes or only the list of document classes that are defined for the current solution when a user adds a document from the local computer as an attachment.
- Open a specified document when the page containing the Attachments widget is opened.

### **IBM Case Manager pages that include this widget by default**

The Attachments widget is included on the following pages:

- Add Task page
- Add Task Form page
- Work Details page
- Work Details Form page
  - “Configuring where attachment content is displayed”
  - “Configuring entry templates for adding and checking in documents” on page 96

*Configuring where attachment content is displayed:*

By default, attachment content is displayed in the Viewer widget that is on the same page as the Attachments widget. Optionally, you can configure the Attachments widget to display content in a separate browser window.

### **About this task**

Attachment content is displayed when the Attachments widget loads and when users double-click attachments or click **View** from the pop-up menu.

**Restriction:** The Attachments widget does not support the following MIME types that are specific to Workplace and Workplace XT. Because the widget does not display these document classes correctly, users might encounter errors when they open the documents of these types. To avoid these errors, store files with the following MIME types in folders that users cannot browse to or that they cannot search for:

#### **Entry template MIME types**

application/x-filenet-customobjectentrytemplate  
 application/x-filenet-declarerecordentrytemplate  
 application/x-filenet-documententrytemplate  
 application/x-filenet-entrytemplate  
 application/x-filenet-folderentrytemplate

#### **IBM Enterprise Records record MIME types**

application/x-filenet-rm-electronicrecord  
 application/x-filenet-rm-emailrecord  
 application/x-filenet-rm-physicalrecord

#### **Search template MIME types**

application/x-filenet-search  
 application/x-filenet-searchtemplate

#### **Workflow-related file MIME types**

application/x-filenet-scenariodefinition

application/x-filenet-workflowdefinition

application/x-filenet-xpdlworkflowdefinition

## Procedure

To configure the Attachments widget:

1. Click the **Edit Settings** icon.
2. Set and save the configuration options:

Option	Description
<b>Open documents in a separate browser window</b>	Displays the content of the attachment in a separate browser window. <b>Important:</b> If a page does not contain the Viewer widget, you must enable this option so that users can open documents from the Attachment widget.
<b>Only show document classes that are defined in the solution</b>	Limits the list of document classes that are displayed when the user adds a document from a local computer as an attachment. When this option is selected, only the document classes that are defined in the solution are displayed. If this option is not selected, users can select from all the document classes in the current object store.
<b>Open a document when the Work Details page is opened</b>	Automatically displays the first or only document in the specified attachment set when the Work Details page or Add Task page opens. The document is displayed in the Viewer widget. <b>Important:</b> Be sure to enter a valid, case-sensitive attachment set name in the <b>Default attachment to display</b> field. If you leave this field blank, provide an invalid attachment set name, or the specified attachment set name doesn't exist for the current work item, the Viewer widget isn't loaded.  The attachment set name is configured in the Case Manager Builder in the <b>Manage Attachments</b> section in the Step Designer. The attachment set name and the actual attachments are configured in Process Designer on the workflow attachment property. <b>Recommendation:</b> Because this widget must work for all case types in this solution, select an attachment set that exists in most, if not all, steps. <b>Restriction:</b> The Viewer widget does not display form content.

## Related information:

Cannot access Process Designer from Case Manager Builder with Google Chrome

*Configuring entry templates for adding and checking in documents:*

You can add controls in the Attachments and Case Information widgets to add and check in documents by using entry templates.

### **Before you begin**

Define entry templates in the IBM Content Navigator Entry Template Manager.

### **About this task**

Entry templates save you time when the items that you are adding or checking in to a repository require the same information. For example, entry templates can specify predefined values, such as the location, class, property values, or security for the items. The information that you enter is consistently applied to all of the items that you add or check in.

**Restriction:** You can create entry templates and configure your case management system to use the entry templates for adding documents to a case. This feature is only supported when the documents that you are adding are stored in the FileNet P8 target object store repository. If your case management system supports adding documents from external repositories, such as IBM Content Manager repositories, entry templates are not supported when you add these documents.

### **Procedure**



To add a control in the Attachment or Case Information widget to check in documents by using an entry template:

1. Click the **Edit Settings** icon for the widget.
2. Select the **Menus** tab and click **Add Menu Item** or select the **Toolbars** tab and click **Add Button**.
3. Select the appropriate action:

<b>Widget</b>	<b>Action</b>
<b>Attachments</b>	Add Document Attachment with an Entry Template
<b>Case Information</b>	Import Document with an Entry Template

4. Specify the version series ID of the entry template to use for adding or importing new documents.

### **Related tasks:**

-  [Creating entry templates in IBM Content Navigator](#)
-  [Creating entry templates in IBM FileNet Workplace XT](#)

### **Calendar widget:**

You can use the Calendar widget to display events for one or more cases. By default, the Calendar widget displays due dates for quick tasks and target end dates for case stages. You can also configure the Calendar widget to display events from an internet calendar that uses the iCalendar file format, such as Google Calendar or Apple Calendar.

You can display the Calendar widget in Case Manager Client by adding the Calendar widget to the Case Details page.

**Tip:** The Calendar widget is supported only on the Case Details page. If you add it to a different page, the widget does not show up at run time.

Alternatively, you can add the **Show Calendar** action to either the Case Toolbar widget or the Case List widget. This action displays the calendar in a dialog box even if you do not add the Calendar widget to a Case page.

If you add the Calendar widget to a page, you can configure the settings to display the calendar in either a small-style or large-style format. Both styles provide visual cues to the user to indicate any events that occur on a day. If a case worker clicks a date in the small-style calendar, IBM Case Manager opens a dialog box with the large-style calendar in day view.

You can subscribe to internet calendars to display events from these calendars in case calendars. In Case Manager Builder, you can configure these subscriptions in the Calendar widget settings for each case type. The events from the internet calendars then display in the calendars for all cases of the specified case type. The URL you enter for a calendar subscription must use a valid iCalendar (.ics) file format.

For the large-style calendar and the calendar that is displayed in the dialog box, case workers can select the **Manage calendar subscriptions** option in Case Manager Client to subscribe to other internet calendars. These subscriptions apply only to the specific case where they are configured.

### **Case Information widget:**

You use the Case Information widget to display an overview of a case to case workers. The Case Information widget can display up to four views: summary, documents, tasks, and history.

The case identifier is displayed at the top of the Case Information widget with a link to open the case in the Case Details page.

### **Case Information widget configuration**

You can edit the settings for the Case Information widget to select the views that are to be displayed in the widget. If you configure the widget to display more than one view, each view is shown on a separate tab. For the Documents view, you can specify whether documents are to open in a separate browser window and whether only document classes that are defined in the solution are to be displayed.

You can enable users to add documents and folders to their Favorites, and to sync content between devices. Favorites and sync must be enabled in IBM Content Navigator before case workers can use the feature.

You can configure toolbars and menus to provide user actions that are specific to the Documents view, History view, and Tasks view. For the Documents view, you can configure toolbars and menus for documents and for folders. You can also enable users to add documents and folders to their Favorites, and to sync the content between devices. Favorites and sync must be enabled in IBM Content Navigator before case workers can use the feature.

You can wire the Case Information widget so that a document is automatically displayed in the Viewer widget when a user selects the document on the Documents tab. To make documents display automatically, wire the outgoing “Document selected” event for the Case Information widget to the incoming “Open document” event for the Viewer widget.

### **IBM Case Manager pages that include this widget by default**

The Case Information widget is included on the following pages:

#### **Add Task page**

By default, the Documents view, History view, and Summary view are displayed on this page.

#### **Add Task Form page**

By default, the Documents view, History view, and Tasks view are displayed on this page.

#### **Cases page**

By default, the History view and Summary view are displayed on this page.

#### **Case Details page**

By default, the Documents view, History view, and Tasks view are displayed on this page.

#### **Case Details Form page**

By default, the Documents view, History view, Summary view, and Tasks view are displayed on this page.

#### **Custom Task Details page**

By default, the Documents view, History view, and Summary view are displayed on this page.

#### **Work Details page**

By default, the Documents view, History view, and Summary view are displayed on this page.

#### **Work Details Form page**

By default, the Documents view, History view, and Summary view are displayed on this page.

### **Case Information widget restrictions**

The Case Information widget does not support the following MIME types that are specific to Workplace and Workplace XT. Because the widget does not display these document classes correctly, users might encounter errors when they open the documents of these types. To avoid these errors, store files with the following MIME types in folders that users cannot browse to or that they cannot search for:

#### **Entry template MIME types**

application/x-filenet-customobjectentrytemplate  
application/x-filenet-declarerecordentrytemplate  
application/x-filenet-documententrytemplate  
application/x-filenet-entrytemplate  
application/x-filenet-folderentrytemplate

#### **IBM Enterprise Records record MIME types**

application/x-filenet-rm-electronicrecord  
application/x-filenet-rm-emailrecord  
application/x-filenet-rm-physicalrecord



### Search template MIME types

application/x-filenet-search  
application/x-filenet-searchtemplate

### Workflow-related file MIME types

application/x-filenet-scenariodefinition  
application/x-filenet-workflowdefinition  
application/x-filenet-xpdlworkflowdefinition

“Task view for the Case Information widget”

“Documents view for the Case Information widget” on page 100

“Displaying extra properties on Documents view” on page 100

“History view for the Case Information widget” on page 101

“Summary view for the Case Information widget” on page 101









*Task view for the Case Information widget:*


You can configure the Case Information widget to provide case workers with a list of the tasks that are defined for the case.

The Task view shows the tasks that you defined for the case type. The tasks are grouped according to whether they are required, optional, or disabled.

In addition to viewing the tasks, case workers can:

- Add tasks to the case.
- Perform actions based on the task state as shown in the following table:

Icon	State	Available user actions
	Waiting	For an optional task that was disabled by the user, the user can enable the task again.
	Disabled Waiting	For a required task that was disabled by the user, the user can enable the task again.
	Ready	The user can start or disable the task.
	Disable Ready	The user can enable or start the task.
	Disabled System	None.
	Working	None.
	Complete	None.
	Failed	None.

Icon	State	Available user actions
	Canceled	None.

In rare circumstances, the task state can be invalid and an error message is displayed for the task. The case worker must contact the system administrator for assistance.

*Documents view for the Case Information widget:*

You can configure the Case Information widget to provide case workers with a list of the documents that are associated with the case.

The Documents view includes a toolbar that contains by default an **Add** button, a **Open** button, and an **Actions** button. The view also provides a pop-up menu that can be used to perform actions on selected documents or folders.

If you configure the IBM Case Manager Client desktop to support external documents, you enable users to add documents from repositories other than the case management repository. These documents can be added as case documents from the Documents tab. When a user adds a document from a repository, the configured repositories are listed.

Typically, searches are not case-sensitive. An exception is object store searches, which are case-sensitive by default. You can change this default search behavior and enable case-insensitive searches for object stores. For more information, see [Setting case-insensitive search behavior for object store searches](#).

*Displaying extra properties on Documents view:*

Document properties are displayed on the Documents tab of the Documents view in the Case Information widget.

### Procedure

To display more properties on the Documents view:

1. Point your browser to the IBM Content Navigator Administration desktop:  
`http://host:port/navigator/?desktop=admin`
2. Log in, then go to the Repositories section.
3. Edit the repository that corresponds to the target object store.
4. Connect to the repository by clicking **Connect**.
5. Go to the Browse tab.
6. Add any additional properties that you want to see in the Documents tab.
7. Save your changes and then click **Close**.
8. Log out of the IBM Content Navigator Administration desktop.
9. Log on to the Case Manager Client desktop to see the properties.

**Important:** The additional properties display only in the Details view. The Magazine view displays only the basic information.

### *History view for the Case Information widget:*

You can configure the Case Information widget to provide case workers with a history of events that occurred for the case. The History view displays a list of the events in chronological order with the most recent events first.

In the History view, a case worker can use preset filter views to easily find relevant case events. A case worker can also select whether detailed information or summary information is to be displayed for each case. The amount of information that is displayed depends upon the type of event. Limited information is displayed for events that are not supported by IBM Case Manager.

For each event, the History view includes the type, title, description, time stamp, and user name.

**Important:** In IBM Case Manager, when selecting to view events from **Earlier this week**, Monday is considered the first day of the week. The History view uses Monday as the start day for a week even if the user locale specifies Sunday as the start day. You cannot change the start day to match the locale.

You can edit the settings for the Case Information widget to specify the number of events that can be displayed on a page in the History view.

### *Summary view for the Case Information widget:*

You can configure the Case Information widget to display a summary of the case properties in Case Manager Builder.

You specify the properties to be displayed in the Summary view by selecting the properties on the **Case Summary** tab for each case type.

#### **Related tasks:**

“Designing views” on page 21

#### **Case List widget:**

You use the Case List widget to display the cases that are returned by a search. Case workers can select a case to view from the list.

The Case List widget displays the results of a search based on the criteria that the case worker entered in the Search widget. The properties that are returned from a search are those properties that were selected for display in the Case Summary view for the case type. These properties which might include:

- The case identifier, which provides a link that the case worker can click to open the case
- The ID of the case worker who last modified the case
- The date that the case was last modified
- The case type
- The case status

You can edit the settings for the Case List widget to:

- Specify the default view for the list and whether case workers can switch between views.
- Show the case health for each case in the list.
- Add a toolbar to the widget.

- Configure the pop-up menu for the widget.

### IBM Case Manager pages that include this widget by default

The Case List widget is included on the Cases page. By default, the pop-up menu for the widget on this page includes the following actions:

- **Add Comment to Case**
- **Create Package**
- **Open Case**
- **Send Link to Case**
- **Show Link to Case**

**Restriction:** The **Create Package** action is included by default only for solutions that are created with version 5.2.1 Fix Pack 4 or later.

### Customized IBM FileNet Content Engine SQL queries

The Case List widget search payload supports customized IBM FileNet Content Engine SQL queries.

For example, you can construct the following Content Engine query to search all cases that were created by admin users (user names that include "admin"):

```
"SELECT [this], t.[FolderName], t.[LastModifier], t.[DateLastModified],
t.[CmAcCaseTypeFolder], t.[CmAcCaseState], t.[CmAcCaseIdentifier],
t.[DateCreated], t.[Creator], t.[Id], t.[ClassDescription] FROM
[CmAcCaseFolder] t WHERE (((t.[creator] LIKE '%admin%' AND
t.[CmAcParentSolution] = Object({CF0CFF8C-CD8C-41D3-AC0A-1272E4F73991})
AND t.[CmAcCaseState] > 0))) ORDER BY t.[CmAcCaseIdentifier]";
```

Custom Content Engine SQL queries must meet the following requirements:

- The SELECT statement must include the following properties: "FolderName", "LastModifier", "DateLastModified", "CmAcCaseTypeFolder", "CmAcCaseState", "CmAcCaseIdentifier", "DateCreated", "Creator", "Id", "ContainerType", "LockToken", "LockTimeout", "ClassDescription",
- The query must search against a Case class, either the CmAcCaseFolder class or its subclass.

To create the search payload object, you can use the `icm.util.SearchPayload` utility class as shown in the following sample code:

```
////////////////////////////////////
// This is running in a script adapter widget
var solution = this.getSolution();
var repositoryId = solution.getTargetOS().id;
var solutionGUID = solution.solutionFolderId;
console.log("solutionFolderId is:" + solutionGUID);
var caseTypeName = "";
var ceQuery = "SELECT t.[FolderName], t.[LastModifier],
t.[DateLastModified], t.[CmAcCaseTypeFolder],
t.[CmAcCaseState], t.[CmAcCaseIdentifier], t.[DateCreated],
t.[Creator], t.[Id], t.[ClassDescription] FROM [CmAcCaseFolder]
t WHERE (((t.[creator] LIKE '%admin%' AND t.[CmAcParentSolution]
= Object(" + solutionGUID + ") AND t.[CmAcCaseState] > 0))) ORDER
BY t.[CmAcCaseIdentifier]";
console.log("ceQuery is:" + ceQuery);

var params = {};
```

```

params.ObjectStore = repositoryId;
params.ceQuery = ceQuery;
params.CaseType = caseTypeName;
params.solution = solution;

var p = new icm.util.SearchPayload();
p.setModel(params);
var payload = p.getSearchPayload();
console.log("Payload is: -----");
console.dir(payload);

return payload;
////////////////////////////////////

```

**Restriction:** Content Engine queries that use the **ceQuery** parameter return one page of results only. The limitation is due to the default page size that is set by IBM Content Navigator (200 rows). For example, if an SQL query matches 500 rows in the Content Engine server, the ceQuery service will return only 200 rows of items.

**Related concepts:**

“Search widget” on page 122

**Related tasks:**

“Designing views” on page 21

**Case Stages widget:**

You use the Case Stages widget to display a visual representation of the stages that a case passes through. The widget shows case workers which stages are complete, which stage is the current stage, and which stages are yet to come.

After you add the Case Stages widget to the Case Details page, you can configure the widget toolbar to enable case workers to perform the following actions:

Action	Description
<b>Complete</b>	Completes the active current stage. The case automatically moves to the next stage. If there is no next stage, all stages are marked as complete. The <b>Complete</b> action cannot be used if the current stage is on hold.
<b>Restart</b>	Restarts the case stage that is previous to the active current stage. If all case stages are complete, the <b>Restart</b> action restarts the last stage. This action cannot be used if the current stage is on hold.
<b>Toggle Hold</b>	Places the active current stage on hold. If the current stage is already on hold, this action returns the stage to the active state.

**Case Toolbar widget:**

You use the Case Toolbar widget to specify the actions that case workers can take for cases. These actions can be implemented as buttons or menu buttons in the toolbar.

The buttons and menu buttons that are included by default in the Case Toolbar widget depend on the type of page that the widget is on.

You can edit the settings for the Case Toolbar widget to:

- Add, remove, and edit the buttons and menu buttons that are available to case workers.
- Show the health of the case in the toolbar.

### **IBM Case Manager pages that include this widget by default**

The Case Toolbar widget is included on the following pages:

#### **Add Case page**

On this page, the Case Toolbar widget contains the following buttons by default:

- **Save Case and Close Page**
- **Close Case Page**

#### **Add Case Form page**

On this page, the Case Toolbar widget contains the following buttons by default:

- **Save Case and Close Page**
- **Close Case Page**

#### **Case Details page**

On this page, the Case Toolbar widget contains the following buttons by default:

- **Add Comment to Case**
- **Add Custom Task** (This menu button includes the following actions: **Add Custom Task** and **Copy Existing Custom Task**.)
- **Add Task**
- **Close Case Page**
- **Create Package**
- **Save Case**
- **Split Case**

**Restriction:** The **Create Package** button is included by default only for solutions that are created with version 5.2.1 Fix Pack 4 or later.

**Tip:** The **Create Package** button opens a wizard that enables users to specify the case artifacts to be included in the package. Use the **Create Default Package** button to create a case package that includes all case artifacts in the package.

#### **Case Details Form page**

On this page, the Case Toolbar widget contains the following buttons by default:

- **Add Comment to Case**
- **Add Custom Task** (This menu button includes the following actions: **Add Custom Task** and **Copy Existing Custom Task**.)
- **Add Task**
- **Close Case Page**
- **Create Package**
- **Save Case**
- **Split Case**

**Restriction:** The **Create Package** button is included by default only for solutions that are created with version 5.2.1 Fix Pack 4 or later.

**Tip:** The **Create Package** button opens a wizard that enables users to specify the case artifacts to be included in the package. Use the **Create Default Package** button to create a case package that includes all case artifacts in the package.

### **Split Case page**

On this page, the Case Toolbar widget contains the following buttons by default:

- **Save Case and Close Page**
- **Close Case Page**

### **Chart widget:**

You can use the Chart widget to display one of the IBM Case Monitor Dashboard charts on a page. These charts provide various views of case-related and task-related statistics for the solution to which a case belongs.

When you configure the settings for the Chart widget, you select the specific chart that you want displayed on the page. For more information about the available charts, see IBM Case Monitor Dashboard pages.

To use the Chart widget, you must configure a Case Analyzer store for the solution's target object store. For more information, see Configuring the Case Analyzer store.

### **Content List widget:**

You use the Content List widget to display a list of documents that are retrieved from the IBM FileNet Content Engine server. Users can use this list to view the documents and their properties, to create document versions, and to download documents.

In the Content List widget, the user can:

- Double-click documents to open them
- Right-click in the Content List widget to perform basic operations, such as viewing, creating document versions, downloading documents, and editing document properties
- Browse through the list of documents

You add the Content List widget to a Work page and then wire the widget to the In-basket widget. Or, you can add the widget to a new page along with the Viewer widget, and then edit the settings for the widget and to specify a stored search URL.

You can edit the settings for the Content List widget to:

- Specify the default view for the list and whether case workers can switch between views
- Indicate how the widget is to retrieve the list of documents that are to be displayed
- Configure a pop-up menu for the documents in the Content List widget  
“Preparing to use the Content List widget” on page 106

“Configuring the list of documents for the Content List widget” on page 107

*Preparing to use the Content List widget:*

To enable users to use the Content List widget, you must add the widget to a page. You then must edit the settings for the Content List widget to indicate how the list of documents is to be generated and, optionally, to configure a pop-up menu.

### **About this task**

You can configure the Content List widget to retrieve a list of documents in one of the following ways:

#### **By using a property list**

The property list contains the IDs of the documents that are to be listed in the Content List widget. To use a property list, you must first create a custom widget that publishes a Display documents event that contains the document IDs.

#### **By running a stored search**

A stored search is a file that contains the search criteria for retrieving the documents. You can create a stored search by creating and saving a search in IBM Content Navigator..

#### **By passing criteria to a stored search and then running the search**

You can wire the Content List widget to substitute values in the stored search criteria. If you do, the widget retrieves and displays documents based on data that it receives from a published event. For example, you can configure the Content List widget to substitute the criteria values that are defined in the stored search. You can substitute these values with work item field values that are published by the In-basket widget or Step Completion widget. You can also substitute values with key-value pairs that are published by a custom widget.

If you want to substitute search criteria with event data at run time, you must set the stored search criteria as editable conditions. If you do not wire the Content List widget to substitute stored search criteria, the widget retrieves documents based on the predefined search criteria.

**Restriction:** The Content List widget does not support certain MIME types that are specific to Workplace and Workplace XT. The widget does not display these document classes correctly and, as a result, users might encounter errors when they open the documents. To avoid these errors, use a property list or a stored search that excludes the following MIME types:

#### **eForm-related types**

application/x-filenet-documentpolicy

application/x-filenet-formdata

application/x-filenet-itxformtemplate

application/x-filenet-workflowpolicy

#### **Entry templates**

application/x-filenet-customobjectentrytemplate

application/x-filenet-declarerecordentrytemplate

application/x-filenet-documententrytemplate



application/x-filenet-entrytemplate  
application/x-filenet-folderentrytemplate  
application/x-filenet-formdataentrytemplate

#### **IBM Enterprise Records records**

application/x-filenet-rm-electronicrecord  
application/x-filenet-rm-emailrecord  
application/x-filenet-rm-physicalrecord

#### **Search templates**

application/x-filenet-search  
application/x-filenet-searchtemplate

#### **Workflow-related files**

application/x-filenet-scenariodefinition  
application/x-filenet-workflowdefinition  
application/x-filenet-xpdlworkflowdefinition

*Configuring the list of documents for the Content List widget:*

You configure the Content List widget to indicate how the widget retrieves the list of documents that are to be displayed.

#### **Procedure**

To configure the Content List widget:

1. Click the **Edit Settings** icon.
2. Click the **Setting** tab.
3. Select the method for populating the Content List widget:

<b>Option</b>	<b>Description</b>
Property list	<p>The Content List widget is to retrieve documents by using the list of document IDs that are contained in the payload of the Display documents event.</p> <p>You must create a custom widget or use the Script Adapter widget to provide the property list to the Content list widget. Wire the event that is published by this widget to the Display documents event that is handled by the Content List widget.</p>

Option	Description
Stored search	<p>The Content List widget is to retrieve documents by running the specified stored search. To specify the stored search:</p> <ol style="list-style-type: none"> <li>1. Select the version of the stored search that you want to run.</li> <li>2. Enter the version series ID of the stored search. You can obtain the identifier by viewing the system properties for the search.</li> </ol> <p>To obtain the version series ID in IBM Content Navigator:</p> <ol style="list-style-type: none"> <li>1. Click the <b>Open Search View</b> icon and then select your target object store.</li> <li>2. Open the folder that contains the stored search file. Right-click the file and click <b>Link &gt; View Link</b>.</li> <li>3. Select and copy the portion of the URL that contains the version series ID. In the URL, <code>vsId=</code> indicates the start of the version series ID.</li> </ol> <p>For example, to select a version series ID, select only the highlighted portion of the following URL:</p> <pre>http://server_name:port/navigator/bookmark.jsp ?desktop=Desktop_Name&amp;repositoryId=repository_ID&amp;repositoryType=p8 &amp;docid=StoredSearchsearch_id&amp;mimeType=application%2Ffx-filenet-search &amp;template_name=StoredSearch&amp;version=released &amp;vsId=%7BF0525657-0000-C622-ADC9-F1CEAD28F92C%7D</pre> <p><b>Tip:</b> Do not include the <code>%7B</code> prefix or <code>%7D</code> suffix when you copy the ID.</p> <p>You can also create a custom widget or use the Script Adapter widget to specify the stored search dynamically at run time. To do so, you configure the custom widget or Script Adapter widget to provide the version series ID and version of the stored search. You then wire the widget to the Search with stored search event that is handled by the Content List widget. This wire overrides the value that is configured for the Content List widget in Case Manager Builder.</p>

### Form widget:

You can use the Form widget instead of the Properties widget to enable workers to view and edit the property values for a case or for a step (work item) by using a form. The form that is used in the Form widget provides a customizable interface and automation features that are not provided by the Properties widget, such as custom field validation and a calculation engine.

The Form widget incorporates forms that are designed in a form application such as IBM Forms. To use the Form widget, you must configure it to use a specific form template (or localization proxy) or to use a form attachment (a form data document, form template, or localization proxy). You can also edit the settings for the Form widget to customize event handling for the widget.

If you use the Add Task Form page or the Work Details Form page, configure the Form widget to use a specific form template. If you use the Form Attachment Work Details page, configure the Form widget to use a form attachment.

**Recommendation:** Configure the Form widget to use an attached form document only when you use the Form widget on the Form Attachment Work Details page.

**Recommendation:** If you plan to use two form widgets in the same column on a page, specify the size of the widget by using either percentage or pixels. Using the automatic size setting can cause unexpected behavior.

You can configure more than one instance of this page to create a page for each step (work item) of a task. Or, you can use the same page for each step, depending on your requirements. If you want to use a different configuration for each step in a task, make copies of the appropriate IBM Case Manager page, and then configure a copy for each step.

You can wire the Form widget to a custom widget so that the custom widget can read and write property values. You can also use the Form widget on a page that contains the Properties widget.

### **IBM Case Manager pages that include this widget by default**

The Form widget is included by default on the following pages:

#### **Add Case Form page**

The Form widget is used to set the initial property values for a new case. For this page, configure the Form widget to use a specific form template.

#### **Case Details Form page**

The Form widget is used to view and modify the property values for a case. For this page, configure the Form widget to use a specific form template.

You can configure more than one instance of this page to create a page for each role. For example, you can configure a page for a case worker and another one with different fields for a manager.

#### **Add Task Form page**

The Form widget is used to view and modify the initial values for step (work item) parameters when you are creating a task. For this page, configure the Form widget to use a specific form template.

#### **Work Details Form page**

The Form widget is used to view and modify the values for step (work item) parameters. For this page, configure the Form widget to use a specific form template.

#### **Form Attachment Work Details page**

The Form widget is used to view and modify the values for step (work item) parameters and to save and attach a form data document to the step after the case worker saves or completes the step.

If you want to use the Form widget to edit a form attachment, you should use the Form Attachment Work Details page instead of the Add Task Form page or the Work Details Form page. This page omits the Attachment widget, which can conflict with the attachment that is used by the Form widget.

For this page, configure the Form widget to use a form attachment.

“Configuring the Form widget to use a form template” on page 110


“Configuring the Form widget to use a form attachment” on page 111

#### **Related concepts:**

“Properties widget” on page 117

#### **Related tasks:**

“Creating a custom page” on page 76

 Integrating forms into your case management application

### Configuring the Form widget to use a form template:

When you configure the Form widget to use a form template, workers can use the widget to create a case or to view and modify the property values for a case or step (work item). The data in the form is mapped to step parameters and case properties. A form data document is not saved.

#### Before you begin

Ensure that a form template is created and saved in the target object store.

**Important:** If you configure the Form widget to use a form template as an interface to collect data, you must store the form template in an IBM FileNet Content Manager object store. You cannot store the form template in IBM Content Manager repository.

#### Procedure

To configure the Form widget to use a form template:

1. Find the object ID or version series ID of the form template as described in the following table and copy it to your Windows clipboard or to a text file. An object ID or version series ID is a unique identification string of 32 hexadecimal characters enclosed by brackets in the following format: {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}. For example, an object ID might be {21484408-092B-4846-A0B8-40D459836147}.

Option	Description
IBM Content Navigator	<ol style="list-style-type: none"><li>1. Click the <b>Open Browse View</b> icon and then select your target object store.</li><li>2. Open the folder that contains the template file. Right-click the file and click <b>Link &gt; View Link</b>.</li><li>3. Select and copy the portion of the URL that contains the object ID or version series ID. In the URL, <code>id=</code> indicates the start of the object ID, and <code>vsId=</code> indicates the start of the version series ID.  For example, to select a version series ID, select only the highlighted portion of the following URL: <pre>http://server_name:port_number/ navigator/bookmark.jsp?desktop=icm &amp;repositoryId=repository_ID&amp; repositoryType=p8&amp;docid=WebFormTemplate %2C%7B6EE7D4D4-928A-4CC5-8CF4-A47EC57362DE%7D %2C%7B80AB1D57-0000-CE19-9B87-209C58077101%7D&amp; mimeType=application%2Fvnd.xfdl.design &amp;template_name=WebFormTemplate&amp;version=released &amp;vsId=%7B80AB1D57-0000-C818-8A54 -797121485D2A%7D</pre></li></ol> <p><b>Tip:</b> Do not include the <code>%2C</code> prefix or <code>%7D</code> suffix when you copy the ID.</p>

Option	Description
<b>IBM Administration Console for Content Platform Engine</b>	<p>To get the object ID for a specific version of a form template:</p> <ol style="list-style-type: none"> <li>1. In the Object Stores directory, open the folder that contains the template file.</li> <li>2. Open the template file.</li> <li>3. Click the <b>Versions</b> tab and open the version for which you want the object ID.</li> <li>4. Click the <b>Properties</b> tab. In the property list, find the ID property and click the ID drop-down arrow.</li> <li>5. Click <b>Display Value</b> and then select and copy the Object ID.</li> </ol> <p>To find the version series ID:</p> <ol style="list-style-type: none"> <li>1. In the Object Stores directory, open the folder that contains the template file.</li> <li>2. Open the template file</li> <li>3. Click the <b>Properties</b> tab. In the property list, find the Version series property.</li> <li>4. Open the property and copy the Version series value.</li> </ol>

2. In Case Manager Builder, open the page that contains the Form widget in Page Designer and click the **Edit Settings** icon on the Form widget.
3. Select **Form template** from the **Open the form by using** list.
4. In the **Version** menu, select the version of the template that you want to use. If you select **Current** or **Released**, the next field requires the version series ID of the template. If you select **Specific**, the next field requires the specific object ID of the template.
5. In the **Version ID** field or the **Version series ID** field, paste the ID that you copied in step 1.
6. Save and deploy the solution.

*Configuring the Form widget to use a form attachment:*

When you configure the Form widget to use a form attachment, workers can use the widget in place of the Properties widget to view and edit the property values for a step (work item). When the worker completes or saves the step, a form data document is saved and attached to the step. You can configure the Form widget to use a form attachment only when you use the Form widget on the Form Attachment Work Details page in the Step Pages space.

### **About this task**

The widget uses a form attachment that is in the target object store. When the step is designed in Case Manager Builder or in Process Designer, the business analyst defines an attachment field. The business analyst can also add an attachment to the field in Process Designer, or a case worker can add the attachment by using the Attachment widget, or the attachment might be added by some other workflow process.

### **Procedure**

To configure the Form widget to use a form attachment:

1. In Case Manager Builder, open the page that contains the Form widget in Page Designer and click the **Edit Settings** icon on the Form widget.

2. Select **Form attachment (step scenarios only)** from the **Open the form by using** list.
3. In the **Folder** field, enter the name of the folder in which to save the form data document. The folder must be a subfolder of the case folder. If the subfolder does not exist, you can create one by entering a new name. If you do not enter a subfolder name, the form data document is saved directly in the case folder.
4. Save and deploy the solution.

**Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

**In-baskets widget:**

You use the In-baskets widget to enable users to view and work with work items.

IBM Case Manager provides three types of in-baskets by default:

**Role-based in-basket**

Lists the work items that are assigned to a specific role. Case workers can double-click work items to open them. The information that is displayed for each work item in the in-basket depends on how the in-basket is configured.

Alternatively, you can configure the role-based in-basket to hide the work items. In this configuration, the case workers does not choose the next work item on which next to work. Instead, the case worker clicks the **Open Next Item** toolbar button to open the next work item in the queue.

**Tip:** If the case worker saves the work item without completing it, the work item is returned to the queue. To ensure that the same case worker can complete the work item, add the **Move Item to Personal In-basket** option to the Work Item Toolbar widget on the Work Details page.

**Personal in-basket**

Lists the work items that are assigned to a specific user. When you configure a role, you can specify whether a personal in-basket is associated with that role. If a personal in-basket is associated with a role, you can opt to use a common personal in-basket for all roles or a role-specific personal in-basket.

**All Assigned Work in-basket**

Lists all the assigned work items. When you configure a role, you specify whether this in-basket is displayed for that role. Typically, you configure this in-basket to be used by a supervisor who monitors that status of all work items.

**Important:** The property values that are in the in-basket are updated only when a case worker completes a work item. The values are not updated if a case worker merely saves changes to the work item.

In Case Manager Builder, you can configure

- One personal in-basket for a solution
- One in-basket for each role in a solution

You can configure additional in-baskets for a role by using Process Designer.

You can edit the settings for the In-baskets widget to:

- Customize the content that is displayed by specifying the role associated with the widget
- Configure a menu or toolbar with custom actions

### IBM Case Manager pages that include this widget by default

The In-baskets widget is included on the Work page.

“Configuring content displayed for the In-baskets widget”

*Configuring content displayed for the In-baskets widget:*

You can configure the content that is displayed in the role-based in-baskets in the In-basket widget.

### About this task

#### Procedure

To configure the In-baskets widget:

1. Click the **Edit Settings** icon.
2. If you want to configure the in-baskets for a specific role, select the role from the **Role** list. If you do not select a role, the changes apply to all roles that use this Work page.
3. Set the following configuration options:

Property	Description
<b>Show work item counts for all in-baskets</b>	Select the check box to display the work item count for all in-baskets that are associated with the role at the first rendering of the In-basket widget.
<b>Hide the tab if there is only one in-basket for the role</b>	Select the check box to hide the tab if only one in-basket is displayed for the selected role or roles.
<b>Do not populate the in-basket until the dynamic filter is received</b>	Select the check box to prevent the work items from being displayed in the in-basket until the filter is received.

4. Select one of the following work modes:

Work Mode	Description
<b>Show work items in role in-baskets</b>	Select the check box to enable the user to view and open the work items in the role in-baskets. This mode gives a user the ability to choose the item to next work on.
<b>Hide work items in role in-baskets</b>	Select the check box to prevent the user from choosing the item to work next on. Instead, an action toolbar is added to the In-basket widget. The user clicks the <b>Open Next Item</b> button on this toolbar to open the next item in the queue. The item is opened in the Work Details page.

5. If you selected the **Show work items in role in-baskets** work mode, configure the following properties:

- a. Indicate if the in-basket will show work items locked by other users by selecting a value from the **Select the locked items to be displayed** list.
- b. Indicate if Case Manager Client users can override the display setting to show or hide work items locked by other users by using the **Allow user to show or hide work items locked by other users** check box.
- c. If you want to prevent users from filtering the work items in the in-basket, select the **Hide the in-basket filter** check box.

The following table describes the result of the different combinations of setting the two properties related to locked items.

<b>Select the locked items to be displayed</b>	<b>Allow user to show or hide work items locked by other users</b>	<b>Result</b>
Work items locked by any user	Selected	When the page is opened, the In-basket widget shows all work items, including those work items that are locked by other users.  The user can hide or show the work items that are locked by other users.
Work items locked by any user	Cleared	When the page is opened, the In-basket widget shows all work items, including those work items that are locked by other users.  The user cannot hide the work items that are locked by other users.
Only work items locked by current user	Selected	When the page is opened, the In-basket widget shows only those work items that are not locked or that are locked by the current user.  The user can hide or show the work items that are locked by other users.
Only work items locked by current user	Cleared	When the page is opened, the In-basket widget shows only those work items that are not locked or that are locked by the current user.  The user cannot show the work items that are locked by other users.

6. Optional: If you selected the **Hide work items in role in-baskets** work mode, provide instructions that tell the user how to open and process the next work item.

#### **Instruction widget:**

You can use the Instruction widget to display the instructions for completing a work item that is associated with a custom task.

Users create the instructions that are displayed in the Instruction widget in the Custom Task Editor. The instructions are in HTML format and can include hyperlinks and property values.

#### **IBM Case Manager pages that include this widget by default**

The Instruction widget is included on the Custom Task Details page.



## Markup widget:

You can use the Markup widget to including custom HTML and JavaScript output on a page in Case Manager Client. For example, you might use this widget to include a personalized message to the case worker who is viewing the page. You might also use this widget to display status information that is generated outside of IBM Case Manager.

The Markup widgets has two modes in the configuration:

### Include markup and scripts

In this mode, you enter the HTML and any JavaScript URLs that are to be used to construct the widget contents.

### Load markup with an event

In this mode, you wire an event from the Script Adapter widget to the Markup widget to construct the widget contents.

“Configuring content for the Markup widget”

“Configuring an event to display content in the Markup widget” on page 116

*Configuring content for the Markup widget:*

You can configure the Markup widget to display content that is generated from HTML, JavaScript URLs, or both.

## Procedure

To configure the Markup widget to display content that is generated from HTML and JavaScript URLs:

1. Click the **Edit Settings** icon.
2. From the **Widget Mode** list, select **Include markup and scripts**.
3. In the **HTML** field, enter the markup that is to be used to construct the widget contents. You can include in the markup any standard HTML tags, valid HTML escape characters, and supported tokens. However, observe the following limitations in the markup:
  - Do not enter text in the format `${abc}`. This format, which is used to load a template or variable with the specified value, is not supported by the Markup widget.
  - The Markup widget does not parse the HTML `<script>` tag. Therefore, you must provide any scripts by using the **JavaScript links** field.

You can use the following tokens in the markup:

Token	Description
<code>#{date}</code>	Displays the current month, day, and year formatted according to the user's locale. For example, the date might be shown as Jul 31, 2017.
<code>#{datetime}</code>	Displays the current month, day, year, and time formatted according to the user's locale. For example, the date and time might be shown as Jul 31, 2017, 11:35 AM.
<code>#{fulldatetime}</code>	Displays the current day of week, month, day, year, and time formatted according to the user's locale. For example, the date and time might be shown as Monday, July 31, 2017 at 11:35 AM.

Token	Description
<code>{rolename}</code>	Displays the role with which the user is associated for the current solution.
<code>{solutionname}</code>	Displays the name of the solution in which this Markup widget is used.
<code>{time}</code>	Displays the current time that is formatted according to the user's locale. For example, the date and time might be shown as 11:35 AM.
<code>{tosname}</code>	Displays the name of the object store that is used by the current solution.
<code>{userid}</code>	Displays the user ID with which the case worker logged in.
<code>{username}</code>	Displays the display name that is defined for the case worker.

- Optional: In the **JavaScript links** field, enter a URL to an external JavaScript file that is used to construct the widget content. To enter multiple URLs, place each URL on a separate line.

*Configuring an event to display content in the Markup widget:*

You can configure the Markup widget to display content that is loaded by an event.

### Procedure

To configure the Markup widget to display content that is loaded by an event:

- Click the **Edit Settings** icon.
- From the **Widget Mode** list, select **Load markup with an event**.
- Add a Script Adapter widget to the page and configure this widget to load the content for the Markup widget.
- Wire the Send event payload outgoing event from the Script Adapter widget to one of the following Markup widget events:

#### **Receive Markup**

Displays the event payload as content in the Markup widget.

#### **Clear Content**

Clears the content from the Markup widget.

### **Original Case Properties widget:**

You use the Original Case Properties widget to display property values for a case that is being split to create a new case. A case worker can compare the values in this widget to the values that are displayed in the Split Case Properties widget for the new case.

The Original Case Properties widget is included by default on the Split Case page so the case worker can view the properties of the original case. The widget is read-only.

### **Process History widget:**

You can use the Process History widget to display the status of the milestones that are defined for the workflow with which a work item is associated.

The Process History widget displays the milestones that the workflow author defines to represent key points in the workflow. The user can track the progress of the workflow by viewing the milestones. For a pending milestone, the widget displays only the milestone name. For a reached milestone, the widget displays the milestone name, an icon, the milestone message, and the date on which the milestone was reached.

You can add the Process History widget to the following pages and the widget automatically handles the events that are broadcast by the other widgets on the page to display milestone information for an opened work item or a selected task:

- Add Task page
- Add Task Form page
- Cases page
- Form Attachment Work Details page
- Work Details page
- Work Details Form page

You can also add the Process History widget to the Work page. On this page, you must manually wire the `icm.SelectWorkItem` event that is published by the In-baskets widget to the `icm.sendWorkItem` event that is handled by the Process History widget.

You can edit the settings for the Process History widget to specify the highest milestone level that is to be visible to users. The Process History widget displays milestones at the specified level and any lower levels. For example, if you enter 5, the Process History widget displays milestones at levels 1, 2, 3, 4, and 5. By default, users see only milestones with level 1.

### **Properties widget:**

You use the Properties widget to enable case workers to view and edit the property values for a case or a work item.

For each case type, IBM Case Manager provides a system-generated layout that you can use for the Properties widget. This layout is automatically generated based on the schema of the properties or data field collection. In the system-generated layout, the property controls are organized vertically in the Properties widget. The properties are listed in alphabetical order. On Work Details pages, work groups are sorted at the top of the list. Attachments are not included.

Alternatively, you can design custom layouts for the Properties widget. You can design these layouts for a case type to meet the requirements of different roles, tasks, or steps. By creating a custom layout, you can determine what properties are included and the order in which the properties are displayed. For example, you might want present a comprehensive set of case properties to analysts, but only a limited set to clerks. You can create one properties layout that includes all the case properties and another to include only a limited number of properties. You then create a Case Details page for analysts and another page for clerks. On each page, you edit the settings for the Properties widget to specify which layout is to be used.

You do not have to configure the properties layout for the Properties widget. Instead, you specify the properties layout that is to be used by default for a case type. This default is used on any page where you have not configured a specific properties layout for a case type.

You can add multiple instances of the Properties widget to a page. You can also add instances of the Properties widget and Form widget to a page.

In the Case Manager Builder Step Designer, you can create workgroups for the steps (work items) in the **Manage Workgroups** section. You can use workgroups to assign work to one or more specific users instead of to any user in a role. In the Case Manager Client, the workgroups are displayed at the top of the Properties widget where you can also add users to the workgroup. The system-generated layout always includes the workgroups. In a custom layout, you can include workgroups to the Properties widget by adding a workflow field to the view.

The number of decimal places (along with autorounding) can be specified. The supported number range is related to the configured number of decimal places. Because a float value can support 15 decimal places, the supported range can be computed as follows:

```
max = 9 * Math.pow(10, 15-places);  
min = -9 * Math.pow(10, 15-places);
```

The Number Text Box and Number Spinner digits automatically enforce these computed ranges.

### **IBM Case Manager pages that include this widget by default**

The Properties widget is included by default on the following pages:

#### **Add Case page**

The Properties widget is used to set the initial property values for a new case.

#### **Add Task page**

The Properties widget is used to set the initial values for step (work item) parameters when you are creating a task.

#### **Case Details page**

The Properties widget is used to view and modify the property values for a case.

#### **Custom Task Details page**

The Properties widget is used to view and modify the values for step (work item) parameters that are defined for a custom task.

#### **Work Details page**

The Properties widget is used to view and modify the values for step (work item) parameters.

### **Content Platform Engine data models**

A case includes the following properties:

- Case properties, which are content data
- Task properties, which are workflow data

Empty property values for different data types are handled differently for content data and workflow data. The model and services layer forces the widget fields to be required if needed.

#### **String properties and datetime properties**

Both content data and workflow data support empty values for string properties and datetime properties. However, an empty value is stored as

null in content data, and an empty value is stored as an empty string in workflow data. The model and services layer for the Properties widget handles these values in the same way for case properties that are displayed in the Case Details page and task properties that are displayed in the Work Details page.

For integer and float properties, the following editors prevent entry of null values if the **Zero when empty** check box is selected:

- **Number text box**
- **Number spinner**

For Boolean properties, the **Check box** editor prevents entry of null values. In addition, the following editors prevent entry of null values if the **False when empty** check box is selected and a zero or false choice is specified:

- **Radio button set**
- **Select**
- **Filtering select**

For workflow data, a timestamp property uses the time of the browser, which is the local time and includes daylight saving time. A time property uses the local time zone that is based on an offset from Coordinated Universal Time (UTC). A time property does not include any adjustments for daylight saving time. If you want users to see the local time including daylight saving time, include the timestamp property on a page. If you include both of these properties on a page, the times they show might not match.

**Important:** For workflow data, datetime property values are not saved as null. Instead, datetime property values are saved with marker values that are equivalent to 1906-08-16 20:26:40Z.

### **Boolean properties**

Content data supports null values for Boolean properties, but workflow data does not. To resolve this difference, the Properties widget does not support empty values for Boolean properties.

**Exception:** The Properties widget supports null values for Boolean properties on Case pages.

The Properties widget displays the field for a Boolean property as a check box. If the check box is selected, the Properties widget stores the value as true. If the check box is not selected, the Properties widget stores the value as false.

If your business solution requires an empty or three-state Boolean value, use a string or integer property with a choice list instead. For example, the choices might be Abstained, Yes, and No.

### **Numeric properties (integer and float)**

Content data supports null values for numeric properties, but workflow data does not. The Properties widget always renders a numeric field as required if the field references the Task property.

On Work Details pages, the Properties widget marks the following fields as required even if you did not define the fields as required in Case Manager Builder:

- Numeric fields and Boolean fields
- Reused numeric properties that are marked as write only for a step.

- Fields that represent a single value that might be part of a choice list in Content Platform Engine. A case property that is defined as a string property with a single value might have a choice list, and the property might be mapped to a workflow step parameter. Because content data accepts a null value, but workflow data does not, the property is marked as required.

The Properties widget marks string fields and datetime fields as required only if you define the fields as required in Case Manager Builder. Because the check box control that is used for Boolean fields automatically enforces a value, the Properties widget does not mark Boolean fields as required.

**Related tasks:**

“Designing views” on page 21

**Script Adapter widget:**

You use a Script Adapter widget to insert logic between widget event communication. to transform the data that is published by one widget into a format understood by another widget. For example, you might use the Script Adapter widget to transform the data that is published by the Properties widget into a format that is understood by a custom widget on a Case Details page. You can also use the Script Adapter widget to insert logic between widget event communication. For example, you might use the Script Adapter to perform custom validation on data that is published by the Properties widget. Because you can configure the Script Adapter widget to display event data at runtime, you can also use the widget to help debug your solution.

When wiring widgets, if the widgets have the same event and the format for the event data is the same, you can simply wire one widget to another. However, if the format for the event data is not the same, you will need to transform the data in the source event into a format expected by the target widget. The Script Adapter widget is the way you achieve this transformation. When the Script Adapter widget receives an event from a widget it is wired to, it displays the event details in the Received Event section. The Script Adapter then runs a script that transforms the data as a function with a "payload" parameter, which is the payload of the incoming event. The script is how you manipulate the payload using any type of logic that you feel is necessary. The value that your custom code returns is the payload of the outbound event of this widget. The Sent Event section displays this information.

For example, a Script Adapter receives a wired event with a payload of "test data", which the Received Event section displays. The Script Adapter has the following script:

```
alert("The value of the payload is: " + payload);  
return "Event Payload: " + payload + "!";
```

The Sent Event section displays "Event Payload: test data!" as the payload for the outbound event.

**IBM Case Manager pages that include this widget by default**

The Script Adapter widget is included on the Cases page by default.

“Inserting logic to transform or validate event data” on page 121

“Debugging events by using the Script Adapter widget” on page 122

*Inserting logic to transform or validate event data:*

You use the Script Adapter widget to insert logic that either transforms or validates event data that is sent by a widget. You use JavaScript code to implement the logic in the Script Adapter widget. The resulting value is contained in the payload of the event that is published by the Script Adapter widget.

### Procedure

To insert logic into event communication on a page:

1. In Case Manager Builder, add the Script Adapter widget to the page.
2. Add a wire to the Script Adapter widget from the source widget that will publish the event whose payload is to be transformed or validated.
3. Add a wire from the Script Adapter widget to the target widget that is to receive the transformed or validated payload.
4. Click the Display menu icon for the Script Adapter widget and then click **Edit Settings**.
5. In the **JavaScript** field, type the code that you want to use to transform or validate the event data. End the script with a Return statement.

The script you enter is limited to basic JavaScript and should be viewed as the body of one single function. You cannot use Dojo commands such as `console.debug()`; you must use the `alert()` statement to display information about the values in the script.

**Tip:** You can use the Script Adapter widget to debug your code while you are developing the script. When it is acting as a debugger, the Script Adapter displays the source event payload so that you can see the data that the script needs to transform.

6. Optional: Hide the Script Adapter widget so that it is not visible to the user.
7. Click **OK**.

### Example

The following example illustrates how you can use the Script Adapter widget to change the type of case to be searched. In this example, the search widget is to be used to locate only cases of the type *Homeowner Policy*.

1. Open the Cases page for the solution in Page Designer.
2. Drag the Script Adapter widget onto the Cases page.
3. From the Case Search widget menu, select **Edit Wiring** to wire the Case Search widget to the Script Adapter widget.
  - a. In the **Script Adapter Incoming Events** section, select **Case Search** from the **Source widget** list.
  - b. Select **Search cases** from the **Outgoing event** list.
  - c. Select **Receive event payload** from the **Incoming event** list.
  - d. Click **Add Wire** and then click **OK**.
4. From the Script Adapter widget menu, select **Edit Settings** and then enter the following script in the **JavaScript** field:

```
payload.CaseType=\"Homeowner Policy\";
alert(\"Updated Case Type to - \" + payload.CaseType);
return payload;
```

This script updates the CaseType property in the payload from the Case Search event.

5. Save the settings and then save and close the page.
6. Deploy the solution.

In Case Manager Client, a search will return only cases of the Homeowner Policy case type.

*Debugging events by using the Script Adapter widget:*

You can use a Script Adapter to view event data to debug problems with wires between two widgets.

### **Procedure**

1. In Page Designer, select **Edit Settings** from Script Adapter widget menu.
2. Select one or both of the following check boxes:

#### **Show Script Text**

Select this check box to display the text of the script as the script is being run.

#### **Block Outbound Event**

Select this check box to prevent the outgoing event from being sent by the Script Adapter widget. If you have wired the *Send event payload* event to another widget, you can select this option to temporarily stop the Script Adapter widget from sending the event while you are debugging the script.

3. Click **OK**.

### **Search widget:**

You use the Search widget to provide case workers with a way to search for cases based on selected property values.

The Search widget provides a basic search and an advanced search. Case workers can use the basic search to find cases based on a specific property value. Basic search searches across all case types. Case workers can use the advanced search to find cases in a specific case type or to narrow the search based on various properties.

You specify the properties that are available in the basic search and in the Advanced Search dialog box by configuring the Search view for the case type.

The Case List widget runs the search by using the criteria that is entered in the Search widget and displays the results.

You can edit the settings for the Search widget to select the case properties that are to be displayed by default in the Advanced Search dialog box.

Typically, searches are not case-sensitive. An exception is object store searches, which are case-sensitive by default. You can change this default search behavior and enable case-insensitive searches for object stores. For more information, see [Setting case-insensitive search behavior for object store searches](#).

### **Case Manager Client pages that include this widget by default**

The Search widget is included on the Cases page.

### **Related concepts:**



“Case List widget” on page 101

**Related tasks:**

“Designing views” on page 21

**Select Case Documents widget:**

You use the Select Case Documents widget to select the documents from an existing case that are also to be associated with a new split case.

The Select Case Documents widget displays a list of the documents that are associated with the original case. You can select from this list those documents that are also to be associated with the new case. Associating the documents with the new case does not remove them from the original case.

**IBM Case Manager pages that include this widget by default**

The Select Case Documents widget is included on the Split Case page by default.

**Split Case Properties widget:**

You can use the Split Case Properties widget to enable case workers to create a new case that reuses property values from an existing case.

The Split Case Properties widget is included on the Split Case page. The widget displays the properties for the case type that is selected for the new case. For properties that match, the values are populated with the values from the original case. The case worker can edit these values as needed for the new case.

The behavior of the Split Case Properties widget is the same as the Properties widget.

**Timeline Visualizer widget:**

The Timeline Visualizer widget provides a visual representation of the extended history for a case. These representations show the progression of events, tasks, and work items over time for the case.

The Timeline Visualizer widget is included on the Case Details page by default. You can also include it on the Work Details page.

The widget uses a horizontal bar, the *timeline*, to show the time duration of each task and work item for the current case. The widget uses vertical bars to represent events, such as filing a document into the current case or changing the properties of the current case. In addition, the widget includes an event density histogram that provides an overview of when in time case events occurred over the entire life of the current case.

By navigating through the events, tasks, and work items, you can use the extended history to determine the status of a specific case. By comparing the extended history for different cases, you can identify potential problems with the workflow that is defined for a case type. For example, you might notice that delays tend to occur during the review step in a particular task. You can then determine an approach to reduce this delay.

**Restriction:** The case history does not include changes to the content or properties of documents in external repositories.

To use the Timeline Visualizer, a case history store must be enabled and configured. Use the IBM Case Manager administration client to configure the extended history. When you configure the extended history, you create the Case History table and select the properties to configure properties auditing. In addition, consider excluding properties whose value is unlikely to change. Excluding such properties reduces clutter in the widget, avoids wasting disk space, and helps performance. Users can still get the values of these properties from the case objects themselves.

By default, the Timeline Visualizer widget initially displays only a high-level timeline that shows the case events. The user can then expand the widget to view an event histogram or the task list for the case. The event histogram shows the density and distribution of case events. The task details can be further expanded to show the work items for each task. The widget switches to full-page mode when it is expanded. You can configure this widget so that the histogram and task list are expanded by default.

The widget initially displays a time range that is based on the number of events that you specify in the widget settings. A user can change the time range by using sliders in the timeline.

By default, the history events are collected every 2 minutes from the server. Therefore, there is a slight delay before the events are displayed in the Timeline Visualizer widget.

### **IBM Case Manager pages that include this widget by default**

The Timeline Visualizer widget is included on the Case Details page by default.

#### **To-Do List widget:**

You use the To-Do List widget to display checklists of to-do items to case workers. To-do items are based on to-do tasks and quick, which do not have an associated workflow. You can define to-do tasks as discretionary if you want to allow case workers to add to-do items at run time. You can enable quick tasks to allow users to create their own to-do tasks.

To-do tasks can be used to record information or track completion of case-related tasks that occur outside of the workflow system. For example, if the solution that you are designing is for automobile claims and one of the case types is for automobile accidents, you can create to-do tasks such as calling the claimant, calling the repair shop, uploading accident photos, and so on.

In Case Manager Builder, you add to-do tasks for case types. Then, you can edit To-Do List widget settings to select the to-do tasks to display in the To-Do List widget. You can also specify which buttons to include on the toolbars for to-do items and the to-do list.

You enable users to create quick tasks by selecting the **Enable case workers to create quick tasks** option for the case type.

#### **Toolbar widget:**

You use the Toolbar widget to enable a case worker to open a web page, add cases, manage roles, or perform a custom action.

The buttons and menu buttons that are included by default in the Toolbar widget depend on the type of page that the widget is on. You can edit the settings for the Toolbar widget to add, remove, and edit the buttons and menu buttons that are available to case workers.

### **IBM Case Manager pages that include this widget by default**

The Toolbar widget is included on the following pages:

#### **Cases page**

On this page, the Toolbar contains the **Add Case** button by default.

#### **Work page**

On this page, the Toolbar contains the following buttons by default:

- **Add Case**
- **Manage Roles**

#### **Viewer widget:**

You use the Viewer widget to display to case workers documents that are stored in FileNet P8 object stores or IBM Content Manager repositories.

The Viewer widget uses the viewer that is configured for IBM Content Navigator to display documents. The IBM Content Navigator viewer allows users to view and annotate image documents.


You cannot configure the Viewer widget from IBM Case Manager. Instead, you use the IBM Content Navigator administration tool to configure a viewer map for the Viewer widget.

### **IBM Case Manager pages that include this widget by default**

The Viewer widget is included on the Work Details page.

“Wiring Viewer widgets to display documents from different widgets”

#### **Related tasks:**

 [Configuring viewers used to display documents in the web client](#)

*Wiring Viewer widgets to display documents from different widgets:*

If you open documents from different widgets on a page, the Viewer widget by default opens each document in a separate tab. To better distinguish the documents that are opened from different widgets, you can place multiple Viewer widgets on the page. You then wire the widgets to display documents in specific Viewer widgets.

#### **Procedure**

In the following steps, assume that a page contains a Content List widget and a Case Information widget with the Documents tab enabled. To display the documents from each widget in a different instance of the Viewer widget, follow these steps:

1. Open the page that contains the Content List widget and Case Information widget in Page Designer.
2. Drag two Viewer widgets onto the page.

**Tip:** There is no label to distinguish the Viewer widgets in Case Manager Client. Therefore, place each Viewer widget on the page to emphasize its relationship to the Content List widget or Case Information widget.

3. Wire the Content List widget to the first Viewer widget on the page:
  - a. Click the **Edit Wiring** icon for the Content List widget.
  - b. In the **Content List Outgoing Events** area, select **Document opened** from the **Outgoing event** list.
  - c. Select the first instance of **Viewer** from the **Target widget** list.
  - d. Select **Open document** from the **Incoming event** list.
  - e. Click **Add Wire**.
  - f. On the **Event Broadcasting** tab, clear the **Enabled** check box for the **Document opened** event. Disabling event broadcasting prevents the Document opened event from being broadcast automatically to the other widgets on the page.
4. Wire the Case Information widget to the second Viewer widget on the page:
  - a. Click **Edit Wiring** icon for the Case Information widget menu.
  - b. In the **Content List Outgoing Events** area, select **Document opened** from the **Outgoing event** list.
  - c. Select the second instance of **Viewer** from the **Target widget** list.
  - d. Select **Open document** from the **Incoming event** list.
  - e. Click **Add Wire**.
  - f. On the **Event Broadcasting** tab, clear the **Enabled** check box for the **Document opened** event. Disabling event broadcasting prevents the Document opened event from being broadcast automatically to the other widgets on the page.
5. Save and deploy the solution.

#### **Website Viewer widget:**

You use the Website Viewer widget to display the website that is associated with a specified URL. For example, you might use this widget to display your company's internal website to case workers. Case workers can browse the website in the Website Viewer widget by clicking the links in the site in the same way that they use a web browser.

You can add the Website Viewer widget to any page that you want.

You edit the settings of the Website Viewer widget to specify the URL for the website that you want to display.

#### **Work Item Toolbar widget:**

You use the Work Item Toolbar widget to enable responding to work items or adding new tasks.

The buttons and menu buttons that are included by default in the Work Item Toolbar widget depend on the type of page that the widget is on. You can edit the settings for the Work Item Toolbar widget to add, remove, and edit the buttons and menu buttons that are available to case workers.

You can also edit the settings for the Work Item Toolbar widget to display the work item name, due date, and instructions in the widget.

## **IBM Case Manager pages that include this widget by default**

The Work Item Toolbar widget is included on the following pages:

### **Add Task page**

On this page, the Work Item Toolbar contains the following buttons by default for adding a task:

- **Start Task and Close Page**
- **Cancel Adding Task**

### **Add Task Form page**

On this page, the Work Item Toolbar contains the following buttons by default for adding a task:

- **Start Task and Close Page**
- **Cancel Adding Task**

### **Custom Task Details page**

On this page, the Work Item Toolbar contains the following buttons by default for processing a work item:

- **Add Comment to Task**
- **Close Work Details Page**
- **Dispatch Work Item**
- **Save Work Item**

### **Form Attachment Work Details page**

On this page, the Work Item Toolbar contains the following buttons by default for processing a work item:

- **Add Comment to Task**
- **Close Work Details Page**
- **Dispatch Work Item**
- **Open Next Work Item**
- **Reassign Item**
- **Save Work Item**

### **Work Details page**

On this page, the Work Item Toolbar contains the following buttons by default for processing a work item:

- **Add Comment to Task**
- **Close Work Details Page**
- **Dispatch Work Item**
- **Open Next Work Item**
- **Reassign Item**
- **Save Work Item**

### **Work Details Form page**

On this page, the Work Item Toolbar contains the following buttons by default for processing a work item:

- **Add Comment to Task**
- **Close Work Details Page**
- **Dispatch Work Item**
- **Open Next Work Item**
- **Reassign Item**

- **Save Work Item**

## **Widget, action, and page events and wiring**

Widgets, actions, and pages communicate by sending outgoing events and receiving incoming events. Outgoing events can be sent over connections that are called *wires* to the incoming events that handle them. Alternatively, outgoing events can be broadcast to be received by any widget that has a corresponding incoming event. Incoming events are those events that a widget or page subscribes to

You can edit wiring and broadcasting of events to control the communication among widgets. For example, you can add the Script Adapter widget to a page and wire it to receive an event from one widget and then transform the event payload before sending the payload to another widget.

“Action events”

“Attachments widget events” on page 140

“Calendar widget events” on page 142

“Case Information widget events” on page 143

“Case List widget events” on page 147

“Case Stages widget events” on page 150

“Case Toolbar widget events” on page 151

“Chart widget events” on page 152

“Content List widget events” on page 153

“Form widget events” on page 156

“In-baskets widget events” on page 159

“Instruction widget events” on page 163

“Markup widget events” on page 164

“Original Case Properties widget events” on page 164

“Page Container widget events” on page 165

“Process History widget events” on page 172

“Properties widget events” on page 173

“Script Adapter widget events” on page 184

“Search widget events” on page 185

“Select Case Documents widget events” on page 186

“Split Case Properties widget events” on page 187

“Timeline Visualizer widget events” on page 197

“To-Do List widget events” on page 199

“Toolbar widget events” on page 202

“Viewer widget events” on page 202

“Website Viewer widget events” on page 203

“Work Item Toolbar widget events” on page 204

### **Action events**

Some of the IBM Case Manager actions publish outgoing events. As with widgets, you can wire these events to the incoming events of widgets.

The action events are listed with the outgoing events that are published by the widget for which the action is configured. For example, the event for the Add Case action is configured by default for the Toolbar widget on the Cases page.

Therefore, the Open new case page event for the Add Case action shows up in the list of outgoing events for the Toolbar widget on this page.

### **Add Case action**

**Action ID**

icm.action.solution.OpenAddCasePage

**Event** Open add case page

**ID** icm.AddCase

**Description**

Opens the Add Case page so that the user can create a case of the selected case type.

**Type** Broadcast

**Payload**

**caseType**

A string that contains the case type that the user selected to create the case.

**caseEditable**

An `icm.model.CaseEditable` object that represents the case that is to be created.

**coordination**

An `icm.util.Coordination` object that is used internally by the widgets in the same page.

### **Add Comment to Case action**

**Action ID**

icm.action.comment.AddCaseComment

**Event** Add case comment

**ID** icm.CommentAdded

**Description**

The new case comment was saved to the case.

**Type** Wired

**Payload**

**Case**

An `icm.model.Case` object that represents the case to which the comment is being added.

**CaseComment**

An `icm.model.CaseComment` object that represents the comment that is being added.

**commentType**

A string that is set to the value "Case" to indicate that the comment applies to a case.

### **Add Comment to Document action**

**Action ID**

icm.action.comment.AddDocumentComment

**Event** Document comment added

**ID** icm.CommentAdded

**Description**

The new document comment was saved to the case.

**Type** Wired

**Payload**

**Case**

An `icm.model.Case` object that represents the case that contains the document for which the comment is being added.

**CaseComment**

An `icm.model.CaseComment` object that represents the comment that is being added.

**commentType**

A string that is set to the value "Document" to indicate that the comment applies to a document.

**docId**

A string that contains the identifier of the document for which the comment is being added.

**documentTitle**

A string that contains the title of the document for which the comment is being added.

### Add Comment to Task action

**Action ID**

`icm.action.comment.AddTaskComment`

**Event** Task comment added

**ID** icm.CommentAdded

**Description**

The new task comment was saved to the case.

**Type** Wired

**Payload**

**Case**

An `icm.model.Case` object that represents the case that contains the task to which the comment is being added.

**CaseComment**

An `icm.model.CaseComment` object that represents the comment that is being added.

**commentType**

A string that is set to the value "Task" to indicate that the comment is being added for a task.

**Task**

An `icm.model.Task` object that represents the task for which the comment is being added.

### Add Comment to Work Item action

**Action ID**

`icm.action.comment.AddWorkItemComment`



**Event** Work item comment added

**ID** icm.CommentAdded

**Description**

The new work item comment was saved to the case.

**Type** Wired

**Payload**

**Case**

An `icm.model.Case` object that represents the case that contains the work item to which the comment is being added.

**CaseComment**

An `icm.model.CaseComment` object that represents the comment that is being added.

**commentType**

A string that is set to the value "WorkItem" to indicate that the comment is being added for a work item.

**WorkItem**

An `icm.model.WorkItem` object that represents the work item for which the comment is being added.

### Add Task action

**Action ID**

`icm.action.case.OpenAddPredefinedTaskPage`

**Event** Open add task page

**ID** icm.AddTask

**Type** Broadcast

**Description**

Opens the Add Task page so the user can add a new discretionary task to the case.

**Payload**

**taskEditable**

An `icm.model.TaskEditable` object that represents the task that is to be created.

**coordination**

An `icm.util.Coordination` object that is used internally by the widgets in the same page.

### Add To-Do Task action

**Action ID**

`icm.action.case.OpenAddToDoTask`

**Event** To-Do Task Added

**ID** icm.ToDoTaskAdded

**Type** Broadcast

**Description**

A discretionary to-do task is added.

**Payload**

### **ToDoTaskEditable**

An `icm.model.TaskEditable` object that represents the added to-do task.

### **Cancel Adding Task action**

#### **Action ID**

`icm.action.task.CancelAddTaskPage`

**Event** Close page

**ID** `icm.ClosePage`

**Type** Broadcast

#### **Description**

Closes the current Add Task page.

#### **Payload**

`null`

### **Close action**

#### **Action ID**

`icm.action.task.CloseToDoTaskView`

**Event** To-Do View Closed

**ID** `icm.ToDoTaskViewClosed`

**Type** Wiring

#### **Description**

The current to-do task view is hidden.

#### **Payload**

##### **caseEditable**

An `icm.model.CaseEditable` object that represents the case that is to be handled.

##### **ToDoTaskEditable**

An `icm.model.TaskEditable` object that represents the added to-do task.

### **Close Case Page action**

#### **Action ID**

`icm.action.case.CloseCasePage`

**Event** Close case page

**ID** `icm.ClosePage`

#### **Description**

Closes the current Add Case, Case Details, or Split Case page without saving any changes.

**Type** Broadcast

#### **Payload**

`null`

## Close Work Details Page action

### Action ID

icm.action.workitem.CloseWorkItemPage

Event Close page

ID icm.ClosePage

### Description

Closes the current Work Details page.

Type Broadcast

### Payload

null

## Complete action

### Action ID

icm.action.task.CompleteToDoTask

Event To-Do Task Completed

ID icm.ToDoTaskCompleted

Type Wiring

### Description

The current to-do task is completed.

### Payload

#### caseEditable

A CaseEditable object that represents the case that is to be handled.

#### ToDoTaskEditable

An icm.model.TaskEditable object that represents the completed to-do task.

## Disable action

### Action ID

icm.action.task.StopToDoTask

Event To-Do Task Disabled

ID icm.ToDoTaskDisabled

Type Wiring

### Description

The current to-do task is disabled.

### Payload

#### caseEditable

An icm.model.CaseEditable object that represents the case that is to be handled.

#### ToDoTaskEditable

An icm.model.TaskEditable object that represents the disabled to-do task.

## Dispatch Work Item

### Action ID

icm.action.workitem.DispatchWorkItemAndClosePage

Event Work item dispatched

ID icm.WorkItemDispatched

Type Wired

### Description

Dispatches the completed work item.

### Payload

#### **workItemEditable**

An icm.model.WorkItemEditable object that represents the work item that is being edited.

Event Close page

ID icm.ClosePage

### Description

Closes the current Work Details page.

Type Broadcast

### Payload

null

## Enable action

### Action ID

icm.action.task.EnableToDoTask

Event To-Do Task Enabled

ID icm.ToDoTaskEnabled

Type Wired

### Description

The current to-do task is enabled.

### Payload

#### **caseEditable**

An icm.model.CaseEditable object that represents the case that is to be handled.

#### **ToDoTaskEditable**

An icm.model.TaskEditable object that represents the enabled to-do task.

## Move Item to Personal In-basket action

### Action ID

icm.action.workitem.MoveToInbox

Event Close Case Page

ID icm.ClosePage

### Description

Closes the case page.

**Type** Broadcast

**Payload**  
null

### **Open action**

**Action ID**  
icm.action.folder.Open

**Event** Folder open

**ID** icm.OpenFolder

**Description**  
Opens the selected folder and displays its content.

**Type** Wired

**Payload**

**folder**

An `icm.model.ContentItem` that represents the folder that was opened.

### **Open Case action**

**Action ID**  
icm.action.case.OpenCasePage

**Event** Open case

**ID** icm.OpenCase

**Description**  
Opens the selected case in the Case Details page.

**Type** Broadcast

**Payload**

**caseEditable**

An `icm.model.CaseEditable` object that represents the case that the user selected.

**coordination**

An `icm.util.Coordination` object that is used internally by the widgets in the same page.

### **Open Item action**

**Action ID**  
icm.action.workitem.OpenWorkItemPage

**Event** Open work item

**ID** icm.OpenWorkItem

**Description**  
Opens the selected work item in the Work Details page.

**Type** Broadcast

**Payload**

**workItemEditable**

An `icm.model.WorkItemEditable` object that represents the work item to be opened.

**coordination**

An `icm.util.Coordination` object that is used internally by the widgets in the same page.

**UIState**

A Dojo Stateful object that is used internally by the widgets in the same page. This object can contain the following properties:

**GetNext**

This property determines whether the next work item is to be opened automatically.

**workitemReadonly**

This property determines whether the work item is to be opened in view mode.

**GetNextCfg**

This property determines the configuration of the Open Next Work Item action.

**Reassign Item action****Action ID**

`icm.action.workitem.Reassign`

**Event** Close case page

**ID** `icm.ClosePage`

**Description**

Closes the Work Details page.

**Type** Broadcast

**Payload**

`null`

**Refresh To-Do List action****Action ID**

`icm.action.case.RefreshToDoTaskList`

**Event** Refresh To-Do List

**ID** `icm.RefreshToDoTaskList`

**Description**

Refreshes the list of to-do tasks in the To-do List widget.

**Type** Broadcast

**Payload (optional)****caseEditable**

An `icm.model.CaseEditable` object that represents the case that is to be displayed.

**coordination**

An `icm.base.Coordination` object that is used internally by the widgets in the same page.

## Restart action

### Action ID

icm.action.task.RestartToDoTask

Event To-Do Task Restarted

ID icm.ToDoTaskRestarted

Type Wiring

### Description

The current to-do task is restarted.

### Payload

#### caseEditable

An icm.model.CaseEditable object that represents the case that is to be handled.

#### ToDoTaskEditable

An icm.model.TaskEditable object that represents the restarted to-do task.

## Save action

### Action ID

icm.action.task.SaveToDoTask

Event To-Do Task Saved

ID icm.ToDoTaskSaved

Type Wiring

### Description

The current to-do task is saved.

### Payload

#### caseEditable

An icm.model.CaseEditable object that represents the case that is to be handled.

#### ToDoTaskEditable

An icm.model.TaskEditable object that represents the saved to-do task.

## Save Case

### Action ID

icm.action.case.SaveCaseOnPage

Event Case saved

ID icm.CaseSaved

### Description

Saves the case that is being edited or added without closing the page.

Type Wired

### Payload

**caseEditable**

An `icm.model.CaseEditable` object that represents the case that is to be saved.

**Save Case and Close Page action****Action ID**

`icm.action.case.AddCaseAndClosePage`

**Event** Case created

**ID** `icm.CaseCreated`

**Description**

Adds the new case to the solution.

**Type** Broadcast

**Payload****caseEditable**

An `icm.model.CaseEditable` object that represents the case that is to be created.

**Event** Close page

**ID** `icm.ClosePage`

**Description**

Closes the current Add Case page or Split Case page.

**Type** Broadcast

**Payload**

`null`

**Save Work Item action****Action ID**

`icm.action.workitem.SaveWorkItemOnPage`

**Event** Work item saved

**ID** `icm.WorkItemSaved`

**Description**

Saves the work item that the user is editing.

**Type** Wired

**Payload****workItemEditable**

An `icm.model.WorkItemEditable` object that represents the work item that is to be saved.

**Split Case action****Action ID**

`icm.action.case.OpenSplitCasePage`

**Event** Open split case page

**ID** `icm.SplitCase`



**Description**

Opens the Split Case page so that the user can reuse properties from an existing case to create a case.

**Type** Broadcast

**Payload****caseType**

A string that contains the case type that the user selected to create the case.

**caseEditable**

An `icm.model.CaseEditable` object that represents the case that is to be split.

**coordination**

An `icm.util.Coordination` object that is used internally by the widgets in the same page.

**Start action****Action ID**

`icm.action.task.StartToDoTask`

**Event** To-Do Task Started

**ID** `icm.ToDoTaskStarted`

**Description**

The current to-do task is started.

**Type** Wired

**Payload****caseEditable**

An `icm.model.CaseEditable` object that represents the case that is to be split.

**ToDoTaskEditable**

An `icm.util.Coordination` object that is used internally by the widgets in the same page.

**Start Task and Close Page action****Action ID**

`icm.action.task.AddTaskAndClosePage`

**Event** Task created

**ID** `icm.TaskCreated`

**Description**

Starts the new task.

**Type** Wired

**Payload****taskEditable**

An `icm.model.TaskEditable` object that represents the task to be started.

**Event** Close page

**ID** `icm.ClosePage`

**Description**








Closes the current Add Task page.

**Type** Broadcast

**Payload**

null

**Related concepts:**

-  [Class icm.model.Case](#)
-  [Class icm.model.CaseEditable](#)
-  [Class icm.model.TaskEditable](#)
-  [Class icm.model.WorkItem](#)
-  [Class icm.model.WorkItemEditable](#)
-  [icm.base.Coordination](#)
-  [icm.model.CaseComment](#)

**Attachments widget events**

The Attachments widget uses events to process the list of documents that are attached to a work item.

For example, the Attachments widget publishes an outgoing event when a user opens a document in the list. The widget handles an incoming event to update the list of documents that are attached to a work item.

“Attachments widget outgoing events”

“Attachments widget incoming events” on page 141

**Attachments widget outgoing events:**

The Attachments widget provides the following outgoing events that are broadcast and wired to the other widgets on the page. The data that is included in the payload of a broadcast event is processed by any widget that has a corresponding handled event. The data that is included in the payload of a wired event is processed by the handled event in the target widget to which the published widget is wired.


**Document opened event**


Description	The user opened a document or an initiating document was opened when the widget loaded.
Event ID	icm.OpenDocument
Type	Broadcast
Payload	<p><b>contentItem</b> An ecm.model.ContentItem object that represents the document that was opened.</p> <p><b>action</b> An ecm.model.Action object that represents the Open action.</p>

### Document selected event

Description	The user selected a document.
Event ID	icm.SelectDocument
Type	Wired
Payload	<b>document</b> An ecm.model.ContentItem object that represents the document that was selected.

### Related concepts:

 Class ecm.model.Action

 Class ecm.model.ContentItem

### Attachments widget incoming events:

The Attachments widget provides incoming events to handle the data that is received from other widgets.

### Add document attachment

Description	Add the document that is contained in the event payload to the attachment that is specified in the event payload.
Event ID	icm.AddAttachment
Payload	<b>attachmentName</b> A string that contains the identifier of the attachment to which the document is to be added.  <b>documents</b> One or more ecm.model.ContentItem objects that represent the documents that are to be added to the specified attachment.

### Clear content event

Description	Clear the content in the Case List widget.
Event ID	icm.ClearContent
Payload	null




### Receive work item

Description	Display the attachments for the work item that is specified in the event payload.
Event ID	icm.SendWorkItem
Payload	<b>workitem</b> An icm.model.WorkItem object that represents the work item for which attachments are to be displayed.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

## Receive new task event

Description	Display the attachment that is associated with the task in the event payload.
Event ID	icm.SendNewTaskInfo
Payload	<b>taskEditable</b> An icm.model.TaskEditable object that represents the task that is to be displayed.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

### Related concepts:

-  Class ecm.model.ContentItem
-  Class icm.model.WorkItem
-  icm.base.Coordination

## Calendar widget events

The Calendar widget handles events to display a calendar that indicates due dates for quick tasks and target end dates for case stages.

“Calendar widget incoming events”

### Calendar widget incoming events:

The Calendar widget provides incoming events to handle the data that is received from other widgets.

## Clear content event

Description	Clear the content in the Calendar widget.
Event ID	icm.ClearContent
Payload	null

## Receive calendar information event

Description	Display the calendar using the data from the payload.
Event ID	icm.SendCaseInfo
Payload	<b>caseEditable</b> An icm.model.CaseEditable object that represents the case for which the calendar is being displayed.

## Refresh Calendar event

Description	Refresh the calendar to update the events that are displayed.
Event ID	icm.RefreshCalendar
Payload	Any value

## Case Information widget events

The Case Information widget handles incoming events to display document, history, task, and summary information about a case.

For example, the Case Information widget handles an incoming event to filter the entries on the History tab based on specified criteria.

“Case Information widget outgoing events”

“Case Information widget incoming events” on page 145

### Case Information widget outgoing events:

The Case Information widget provides the following outgoing events that are broadcast and wired to the other widgets on the page. The data that is included in the payload of a broadcast event is processed by any widget that has a corresponding handled event. The data that is included in the payload of a wired event is processed by the handled event in the target widget to which the published widget is wired.

#### Document opened event

Description	The user opened a document
Event ID	icm.OpenDocument
Type	Broadcast
Payload	<b>contentItem</b> An <code>ecm.model.ContentItem</code> that represents the document that was opened.  <b>action</b> A string that indicates whether the document was opened for viewing or for editing. The valid values are <code>view</code> or <code>edit</code> .

#### Document selected event

Description	The user selected a document.
Event ID	icm.SelectDocument
Type	Wired
Payload	<b>contentItem</b> An <code>ecm.model.ContentItem</code> that represents the document that was selected.

#### Folder opened event

Description	The user opened a folder
Event ID	icm.OpenFolder
Type	Wired
Payload	<b>contentItem</b> An <code>ecm.model.ContentItem</code> that represents the folder that was opened.

### Folder selected event

Description	The user selected a folder.
Event ID	icm.SelectFolder
Type	Wired
Payload	<b>contentItem</b> An <code>ecm.model.ContentItem</code> that represents the folder that was selected.

### Task selected event

Description	The user selected a task from the Tasks view. If the Case Information widget is wired to the Process History widget, the event payload includes the milestones for the selected task.
Event ID	icm.TaskSelect
Type	Wired
Payload	<b>task</b> An <code>icm.model.Task</code> object that represents the task that the user selected.

### Send case history event

Description	The user selected a history item from the History view.
Event ID	icm.SendCaseHistory
Type	Wired
Payload	<b>caseHistory</b> An object that represents the item the user selected in the History view. <b>contentItem</b> For a document history item only: An <code>ecm.model.ContentItem</code> that represents the document that was selected. <b>action</b> For a document history item only: A string containing the value "open" to indicate that the document is to be opened in the Viewer widget.

### Task status updated event



Description	The task status is updated.
Event ID	icm.TaskStatusUpdated
Type	Wired

---

Payload	<p><b>action</b> A string that indicates the action that the user performed on the task. For example, the action might be start, disable, enable, or so on.</p> <p><b>taskObj</b> An <code>icm.model.Task</code> object that represents the task on which the user performed an action.</p>
---------	---

---

**Related concepts:**

-  [Class `ecm.model.ContentItem`](#)
-  [Class `icm.model.Task`](#)

**Case Information widget incoming events:**

The Case Information widget provides incoming events to handle the data that is received from other widgets.

**Clear content event**

Description	Clear the content in the Case Information widget.
Event ID	<code>icm.ClearContent</code>
Payload	<code>null</code>

**Filter History event**

Description	Filter the entries on the History tab based on the criteria that is specified in the event payload.
Event ID	<code>icm.FilterHistory</code>
Payload	<p><b>show</b> A string that contains the label for the filter that determines how much information is displayed in the History tab. By default, the values are “Summary” and “All”.</p> <p><b>showValue</b> A string that contains the value for the filter that determines how much information is displayed in the History tab. The valid values are <code>Summary</code> and <code>All</code>.</p> <p><b>forEntry</b> A string that contains the label for the filter that determines the category of items to be displayed in the History tab. The valid values are “All”, “Case”, “Comments”, “Documents”, “Tasks”, and “Folders”.</p> <p><b>forValue</b> A string that contains the filter that determines the category of items to be displayed in the History tab. The valid values are <code>All</code>, <code>Case</code>, <code>Comments</code>, <code>Documents</code>, <code>Tasks</code>, and <code>Folders</code>.</p>

---

### Receive new task event

Description	Display the case information for the case that is related to the task in the event payload.
Event ID	<code>icm.SendNewTaskInfo</code>
Payload	<b>taskEditable</b> An <code>icm.model.TaskEditable</code> object that represents the task that is to be added. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

### Receive work item event

Description	Display the case information for the case that is related to the work item in the event payload.
Event ID	<code>icm.SendWorkItem</code>
Payload	<b>workItemEditable</b> An <code>icm.model.WorkItemEditable</code> object that represents the work item for which case information is to be displayed.

### Refresh event

Description	Refresh the tab that is specified in the event payload.
Event ID	<code>icm.RefreshTab</code>
Payload	<b>tabId</b> A string containing the identifier of the tab that is to be refreshed. The valid values are: Documents, History, Summary, and Tasks.

### Select Case event

Description	Display the case information for the case that is contained in the event payload.
Event ID	<code>icm.SelectCase</code>
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be displayed.

### Select folder event

Description	Open the case subfolder that is specified in the event payload.
Event ID	<code>icm.SelectInitialFolder</code>
Payload	<b>folderPath</b> A string containing the path to the selected case subfolder.



### Select tab event

Description	Open the tab that is specified in the event payload.
Event ID	icm.SelectTab
Payload	<b>tabId</b> A string containing the identifier of the tab that is to be opened. The valid values are: Documents, History, Summary, and Tasks.

### Send Case Information event

Description	Display the case that is specified in the event payload.
Event ID	icm.SendCaseInfo
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be displayed. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

#### Related concepts:

- [↗](#) Class `icm.model.CaseEditable`
- [↗](#) Class `icm.model.TaskEditable`
- [↗](#) Class `icm.model.WorkItemEditable`
- [↗](#) `icm.base.Coordination`

### Case List widget events

The Case List widget uses events to process the list of cases that are returned when a user searches for cases.

For example, the Case List widget publishes an outgoing event to open a case that a user double-clicks. The Case List widget handles an incoming event to display the list of cases that were returned by a search.

“Case List widget outgoing events”

“Case List widget incoming events” on page 148

#### Case List widget outgoing events:

The Case List widget provides outgoing events that are broadcast to the other widgets on the page. The data that is included in the payload of a broadcast event is processed by any widget that has a corresponding handled event.

### Open case event

Description	The user selected a case that is to be opened in the Case Details page.
Event ID	icm.OpenCase
Type	Broadcast

---

Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that the user selected.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>
---------	--

---



### Select case event

---

Description	The user selected a case in the list or no case is selected in the list. If one or more cases are selected, the event is broadcast for the first case selected by the user.
Event ID	<code>icm.SelectCase</code>
Type	Broadcast
Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the first case that the user selected.</p> <p>The payload is null if no case is selected.</p>

---

#### Related concepts:

-  [Class `icm.model.CaseEditable`](#)
-  [icm.base.Coordination](#)

#### Case List widget incoming events:

The Case List widget provides incoming events to handle the data that is received from other widgets.

#### Clear content event

---

Description	Clear the content in the Case List widget.
Event ID	<code>icm.ClearContent</code>
Payload	null

---

#### Refresh event

---

Description	Rerun the current search query and refresh the list of cases.
Event ID	<code>icm.RefreshCaseList</code>
Payload	null

---

#### Search cases event

---

Description	Update the case list with the cases that were returned by a search.
Event ID	<code>icm.SearchCases</code>

---

---

## Payload

### **searchTemplate**

An `ecm.model.SearchTemplate` object that represents the search criteria. The Case List widget can run the search directly by using this template. This parameter is ignored if a value is provided for the `ceQuery` parameter.

### **searchProperties**

An array of properties that are used by the search criteria.

### **caseType**

The symbolic name of the case type that is being searched. If the search is across a solution, the value of this parameter is "".

### **caseTypeTitles**

A JSON object that contains the symbolic name of the case title property for each case type that is being searched. The structure is:

```
caseTypeTitles: {  
  caseType1: caseTitle symbolic name1,  
  caseType2: caseTitle symbolic name2  
}
```

### **detailsViewProperties**

An array of properties that are to be displayed in the details view of the Case List widget.

### **magazineViewProperties**

A JSON object that contains an array of properties for each case type that is being searched. These properties include the Case Summary View properties and well-known properties (`CmAcmCaseIdentifier`, `CmAcmCaseState`, `CmAcmCaseTypeFolder`, `LastModifier`, `DateLastModified`).

The structure is of the JSON object is:

```
magazineViewProperties: {  
  caseType1: [prop1, prop2, prop3],  
  caseType2: [prop1, prop2, prop3],  
}
```

These properties are displayed in the second line for each case in the magazine view.

### **ceQuery**

A user-provided Content Engine query statement that is used to search cases. If a value is specified for this parameter, the `searchTemplate` parameter is ignored.

---

## Select a row event

---

Description	Select a row by the row number or the case identifier.
-------------	--

---

Event ID	<code>icm.SelectRow</code>
----------	----------------------------

---

Payload	The payload can contain one of the following values:
---------	--

---

### **caseID**

A string that contains the identifier of the case that is to be selected.

### **rowNumber**


A string that contains the number of the row that is to be selected.


---

## Sort by property event

Description	Sort the list of cases by the specified property.
Event ID	icm.SortByProperty
Payload	<b>symbolicName</b> A string that contains the symbolic name of the column that is to be used to sort the case list.

### Related concepts:

 [Class icm.model.CaseEditable](#)

 [Class icm.model.Solution](#)

## Case Stages widget events

The Case Stages widget handles events to display the current state of the stages for a case.

“Case Stages widget incoming events”

### Case Stages widget incoming events:

The Case Stages widget provides incoming events to handle the data that is received from other widgets.

## Complete stage event

Description	Complete the current stage.
Event ID	icm.CompleteStage
Payload	null

## Open case event

Description	Display the stages for the case that is being opened.
Event ID	icm.SendCaseInfo
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case for which the stages are to be displayed. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

## Receive work item event

Description	Display the stages for the case that is related to the work item in the event payload.
Event ID	icm.SendWorkItem
Payload	<b>WorkItemEditable</b> An <code>icm.model.WorkItemEditable</code> object that represents the work item that is related to the case for which the stages are to be displayed.

### Refresh stages event

Description	Refresh and redisplay the stages from the current case.
Event ID	icm.RefreshStage
Payload	Any value

### Restart stage event

Description	Restart the previous stage.
Event ID	icm.RestartStage
Payload	null

### Select case event

Description	Display the stages for the case that is currently selected.
Event ID	icm.SelectCase
Payload	<b>caseEditable</b> An icm.model.CaseEditable object that represents the case for which the stages are to be displayed.

### Stage changed event

Description	The current stage has been changed.
Event ID	icm.StageChanged
Payload	null

### Toggle the current stage event

Description	Place the stage to the on-hold state if the stage is currently working. Place the stage to the working state if the state is currently on hold.
Event ID	icm.ToggleOnHoldStage
Payload	null

## Case Toolbar widget events

The Case Toolbar widget handles events to display and process actions that the user can perform for a case.

For example, the Case Toolbar widget handles an incoming event to display the properties for a case type so that the user can create a case.

The Case Toolbar widget also publishes events for certain actions that you add to the widget.

“Case Toolbar widget incoming events”

### Case Toolbar widget incoming events:

The Case Toolbar widget provides incoming events to handle the data that is received from other widgets.

### Create case event

Description	Display the properties for the specified case type so the user can create a case.
Event ID	<code>icm.SendNewCaseInfo</code>
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be created. <b>caseType</b> An <code>icm.model.CaseType</code> object that represents the type of case that is to be created. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

### Open case event

Description	Open the case that is specified in the event payload.
Event ID	<code>icm.handleSendCaseInfoEvent</code>
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be opened. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

### Split case event

Description	Display the information for the original case that is being split to create a new case.
Event ID	<code>icm.SendSplitCaseInfo</code>
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be split. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

#### Related concepts:

 [Class `icm.model.CaseEditable`](#)

### Chart widget events

The Chart widget handles events to display a chart that provides business activity data that is specific to your case management system.

For example, the Chart widget handles incoming events to display either case-related or task-related statistics.

“Chart widget incoming events” on page 153

### Chart widget incoming events:

The Chart widget provides incoming events to handle the data that is received from other widgets.

#### Clear content event

Description	Clear the content in the Chart widget.
Event ID	icm.ClearContent
Payload	null

#### Open work item event

Description	Display task-related data from the payload in the chart that is configured for the Chart widget.
Event ID	icm.SendWorkItem
Payload	<b>workItemEditable</b> An icm.model.WorkItemEditable object that represents the work item for which case information is to be displayed.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

#### Send case information event

Description	Display case-related data from the payload in the chart that is configured for the Chart widget.
Event ID	icm.SendCaseInfo
Payload	<b>caseEditable</b> An icm.model.CaseEditable object that represents the case that is to be displayed.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

### Content List widget events

The Content List widget uses events to display a list of documents from which users can access documents, create document versions, and download documents.

For example, the Content List widget publishes an outgoing event when a user selects a document in the list. The Content List widget handles an incoming event to run a stored search for documents.

“Content List widget outgoing events”

“Content List widget incoming events” on page 154

#### Content List widget outgoing events:

The Content List widget provides outgoing events that are wired to the other widgets on the page. The data that is included in the payload of a wired event is processed by the handled event in the target widget to which the published widget is wired.


### Document opened event

Description	The user opened a document
Event ID	<code>icm.OpenDocument</code>
Type	Broadcast
Payload	<b>contentItem</b> An <code>ecm.model.ContentItem</code> that represents the document that was opened.

### Document selected event

Description	The user selected a document
Event ID	<code>icm.SelectDocument</code>
Type	Wired
Payload	<b>contentItem</b> An <code>ecm.model.ContentItem</code> that represents the selected document.

#### Related concepts:

 [Class `ecm.model.ContentItem`](#)

#### Content List widget incoming events:

The Content List widget provides incoming events to handle the data that is received from other widgets.

#### Clear content event

Description	Clear the content in the Content List widget.
Event ID	<code>icm.ClearContent</code>
Payload	<code>null</code>

#### Display documents event

Description	Display the list of documents that are referenced in the event payload.
Event ID	<code>icm.ReceivedDocumentIDs</code>



---

**Payload****objectStoreNames**

An array that contains the IBM Content Navigator repository IDs of the target object stores.

**symbolicNames**

An array that contains the symbolic names for the Document properties.

**values**

An array of JSON objects that contains the versions series identifiers for the documents.

**externalColumns**

An array of Dojo objects in which each object contains the name and identifier of an external column.

The following example illustrates the payload:

```
payload = {  
  "objectStoreNames" : ["TOS"],  
  "symbolicNames": [...],  
  "values": [...],  
  "externalColumns": [{"symbolicName": "", "name": ""}]  
};
```

---

**Search with stored search event**

---

Description	Run the stored search that is specified in the payload and display the list of documents that are returned by the search.
Event ID	icm.ReceivedStoredSearch
Payload	<b>StoredSearch.objectStoreName</b> The IBM Content Navigator repository ID of the target object store or the target object store name. <b>StoredSearch.vsId</b> The version series identifier of the stored search that is to be run. <b>StoredSearch.version</b> A string that indicates the version of the stored search that is to be run. The valid values are current or released.

---

**Search with values event**

---


Description	Use the name and value pairs that are specified in the event payload as the criteria for the configured stored search. Run the search and display the list of documents that is returned.
Event ID	icm.ReceivedSearchValues
Payload	An array of JSON objects in which each object contains a property name and value pair that is used as criteria for the configured stored search.: <b>name</b> A string that contains the symbolic name of the property. <b>value</b> A string that contains the value of the property.

---

## Search with work item event

Description	Use the workflow data field values that are specified in the event payload as the criteria for the configured stored search. Run the search and display the list of documents that is returned.
Event ID	icm.SendWorkItem
Payload	<b>workItem</b> An icm.model.WorkItem object that represents the work item for which a list of documents is to be returned.

### Related concepts:

 [Class icm.model.WorkItem](#)

## Form widget events

The Form widget publishes and handles events to display case and work item properties in a form for users to view and edit.

For example, the Form widget handles an incoming event to display the properties for a case type so that the user can create a case of that type.

“Form widget outgoing events”

“Form widget incoming events” on page 157

### Form widget outgoing events:

The Form widget provides outgoing events that are wired to the other widgets on the page. The data that is included in the payload of a wired event is processed by the handled event in the target widget to which the published widget is wired.

## Field updated event

Description	The value of a field was updated.
Event ID	icm.FieldUpdated
Type	Broadcast

---

Payload

**changes**

A JSON object that contains the unique identifier and value for the property that was modified. The structure is:

```
changes: [  
  {  
    id: prop1,  
    type: integer,  
    array: false,  
    value: value1  
  }  
]
```

**Important:** The type parameter for the changes JSON object is deprecated.

The id property can be the name of a form field, even if that field is not mapped to a case property. For IBM Forms, the form field must be published.

The data type of the value corresponds to the data type of the property, or is null. For properties that support multiple values, the value is an array of the corresponding data type.

---

**Form widget incoming events:**

The Form widget provides incoming events to handle the data that is received from other widgets.

**Update field event**

---

Description	Update the fields with the values that are contained in the event payload.
-------------	--

---

Event ID	icm.UpdateFields
----------	------------------

---

Payload

**changes**

A JSON object that contains the unique identifier and value for each property that is to be modified. The structure is:

```
changes: [  
  {  
    id: prop1,  
    type: integer,  
    array: false,  
    value: value1  
  }  
]
```

**Important:** The **type** parameter for the **changes** JSON object is deprecated.

The id property can be the name of a form field, even if that field is not mapped to a case property. For IBM Forms, the form field must be published.

The data type of each value must correspond to the data type of the property, or must be null. For properties that support multiple values, the value is an array of the corresponding data type. If the value is incompatible, the value is coerced if possible or a default value is applied. A warning is also logged.

---

### Clear content event

---

Description	Clear the content in the Form widget.
Event ID	icm.ClearContent
Payload	null

---

### Send case information event

---

Description	Display the properties for the specified case in the form. Replace the existing form if one exists.
Event ID	icm.SendCaseInfo
Payload	<b>caseEditable</b> An icm.model.CaseEditable object that represents the case that is to be displayed and updated.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

---

### Send new case information event

---

Description	Display the properties for the new case that the user is creating.
Event ID	icm.SendNewCaseInfo
Payload	<b>caseEditable</b> An icm.model.CaseEditable object that represents the case that is to be created.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.  Alternatively, the payload can be null.

---

### Send new task information event

---

Description	Display the properties for the new task that the user is creating.
Event ID	icm.SendNewTaskInfo
Payload	<b>taskEditable</b> An icm.model.TaskEditable object that represents the task that is to be created. The editable properties are contained in the icm.model.LaunchStep object that is associated with the icm.model.TaskEditable object. You can obtain the editable properties by calling the taskEditable.getLaunchStep() method.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

---

## Send work item event

Description	Display the properties for the work item that is contained in the event payload.
Event ID	icm.SendWorkItem
Payload	<b>workItemEditable</b> An icm.model.WorkItemEditable object that represents the work item that is to be displayed and updated.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.  Alternatively, the payload can be null.

### Related concepts:

- [↗](#) Class icm.model.CaseEditable
- [↗](#) Class icm.model.TaskEditable
- [↗](#) Class icm.model.WorkItemEditable
- [↗](#) icm.base.Coordination

## In-baskets widget events

The In-baskets widget uses events to display and process lists of the work items that are assigned to a user.

For example, the In-baskets widget publishes an outgoing event when the user selects a work item in an in-basket to open. The In-baskets widget handles an incoming event to apply user-specified filters to the list of work items.

“In-baskets widget outgoing events”

“In-baskets widget incoming events” on page 161

### In-baskets widget outgoing events:

The In-baskets widget provides outgoing events that are broadcast and wired to the other widgets on the page. The data that is included in the payload of a broadcast event is processed by any widget that has a corresponding handled event. The data that is included in the payload of a wired event is processed by the handled event in the target widget to which the published widget is wired.

### In-basket selected event

Description	The user opened the page that contains the In-baskets widget or has clicked a tab to display a different in-basket.
Event ID	icm.SelectInbasket
Type	Wired
Payload	<b>selectInbasket</b> An ecm.model.ProcessInbasket object that represents the in-basket that is to be displayed.

## Open work detail page event

---

Description	The user selected a work item to open.
Event ID	icm.OpenWorkItem
Type	Broadcast
Payload	<p><b>workitem</b> An <code>icm.model.WorkItemEditable</code> object that represents the selected work item.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p> <p><b>UIState</b> A Dojo Stateful object that is used internally by the widgets in the same page. This object can contain the following properties:</p> <p><b>GetNext</b> This property determines whether the next work item is to be opened automatically.</p> <p><b>workitemReadonly</b> This property determines whether the work item is to be opened in view mode.</p> <p><b>GetNextCfg</b> This property determines the configuration of the Open Next Work Item action.</p>

---

## Work item selected event

---

Description	The user clicked a row in the in-basket to select the work item.
Event ID	icm.SelectWorkItem
Type	Wired
Payload	<p><b>workitem</b> An <code>icm.model.WorkItemEditable</code> object that represents the selected work item.</p>

---

## Send selected work items event

---

Description	The In-basket widget received the <code>icm.RequestSelectedWorkItems</code> event that requested information for specified work items.
Event ID	icm.SendSelectWorkItems
Type	Wired

---

---

Payload

**result**

The payload that was passed by the `handleRequestSelectedWorkItems` function.

**selectWorkItems**

An array of `icm.model.WorkItem` objects that represent the work items in the specified in-basket rows.

---

**Related concepts:**

- [↗](#) Class `ecm.model.ProcessInbasket`
- [↗](#) Class `icm.model.WorkItem`
- [↗](#) Class `icm.model.WorkItemEditable`
- [↗](#) `icm.base.Coordination`

**In-baskets widget incoming events:**

The In-baskets widget provides incoming events to handle the data that is received from other widgets.

**Apply filter event**

---

Description	Update the work items that are listed in the in-basket based on the specified filters.
Event ID	<code>icm.ApplyFilter</code>

---

Payload	<p>The Apply filter event can accept either of the following payloads:</p> <p><b>filters</b>  An array of <code>icm.model.InbasketFilter</code> objects that represent the filters that are to be applied to the in-basket. This array uses the following format:</p> <pre>var filter = icm.model.InbasketFilter.fromJSON(dynamicFilterJSON); var filters = []; filters.push(filter); var payload = {"filters": filters};</pre> <p>The following example shows the contents of this payload:</p> <pre>var filterJSON = {...}; var filter = icm.model.InbasketFilter.fromJSON(filterJSON); var filters = []; filters.push(filter); var payload = {"filters": filters};</pre> <p><b>dynamicFilters</b>  An array of <code>icm.model.InbasketDynamicFilter</code> objects that represent the filters that are to be applied to the in-basket. This array uses the following format:</p> <pre>var payload =   {"dynamicFilters": dynamicFilters,    "cursorLocation": 1,    "cleanDynamicFilterByReset":true};</pre> <p>The optional <code>cursorLocation</code> parameter specifies the default cursor location. The optional <code>cleanDynamicFilterByReset</code> parameter specifies whether a dynamic filter is cleared when the user clicks the <b>Reset</b> button.</p> <p>The following code provides an example of this payload:</p> <pre>var payload =   {"dynamicFilters": dynamicFilters,    "cursorLocation": 1,    "cleanDynamicFilterByReset":true};</pre>
---------	--

---

### Clear content event

Description	Clear the content in the In-baskets widget.
Event ID	<code>icm.ClearContent</code>
Payload	<code>null</code>

### Receive In-basket event

Description	Display the tab for the in-basket that is contained in the event payload.
Event ID	<code>icm.ReceiveInbasket</code>
Payload	<p><b>queueName</b>  The name of the queue in which the work items in the in-basket are stored.</p> <p><b>inbasketName</b>  The name of the in-basket that is being received.</p> <p><b>cursorLocation</b>  Optional: The index of the row that is to be selected in the in-basket.</p>

---



### Receive Role event

Description	Update the In-baskets widget to display the in-baskets that are associated with the specified role.
Event ID	icm.ReceiveRole
Payload	<b>role</b> A string that contains the symbolic name of the role.

### Refresh event

Description	Refresh the in-basket to update the list of work items..
Event ID	icm.Refresh
Payload	Any value

### Request open work item event

Description	Send the information to open the next work item in the queue.
Event ID	icm.RequestOpenWorkItem
Payload	<b>queueName</b> The name of the queue in which the work item to be opened is stored. <b>inbasketName</b> The name of the in-basket in which the work item to be opened is located. <b>cursorLocation</b> Optional: The index of the row in the in-basket that contains the work item to be opened.

### Request selected work items event

Description	Send the information for the in-basket rows that are specified in the event payload.
Event ID	icm.RequestSelectWorkItems
Payload	Any. The In-basket widget handles the event and sends the specified work items along with the incoming payload.

#### Related concepts:

 [icm.model.DynamicFilter](#)

#### Related information:

[handleApplyFilter](#)

### Instruction widget events

The Instruction widget handles an incoming event to display the instructions that were defined for a custom task.

“Instruction widget incoming events”

#### Instruction widget incoming events:

The Instruction widget provides an incoming event to handle the data that is received from other widgets.

## Send work item event

Description	Display the properties for the work item that is contained in the event payload.
Event ID	icm.SendWorkItem
Type	Wired
Payload	<b>workItemEditable</b> An icm.model.WorkItemEditable object that represents the work item to be opened.

### Related concepts:

 Class icm.model.WorkItemEditable

## Markup widget events

The Markup widget handles incoming events from the Script Adapter widget to construct the content that is displayed.

“Markup widget incoming events”

### Markup widget incoming events:

The Markup widget provides incoming events to handle the data that is received from the Script Adapter widget.

## Clear Content event

Description	Clear the content from the Markup widget.
Event ID	icm.ClearContent
Payload	null

## Receive Markup event

Description	Display the content is contained in the event payload.
Event ID	icm.ReceiveMarkup
Payload	<b>markupText</b> The HTML tags and text that is used to construct the contents of the Markup widget.

## Original Case Properties widget events

The Original Case Properties widget handles incoming events to display the properties that are defined for case that is being split.

“Original Case Properties widget incoming events”

### Original Case Properties widget incoming events:

The Original Case Properties widget provides incoming events to handle the data that is received from other widgets.

## Clear content event


Description	Clear the content in the Case List widget.
Event ID	icm.ClearContent

Payload	null
---------	------

### Original case event

Description	Display the properties of an existing case from which property values are to be reused in a new case.
Event ID	icm.SendSplitCaseInfo
Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be created. This object provides the properties of the original case by using the <code>getSplitSource()</code> method.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>

### Related concepts:

 [Class `icm.model.CaseEditable`](#)

 [icm.base.Coordination](#)

### Page Container widget events

The Page Container widget uses events to control the overall behavior of a page.

For example, the Page Container widget publishes an outgoing event when the user closes the page. The Page Container widget handles incoming events to collapse and expand regions on a page.

“Page Container widget outgoing events”

“Page Container widget incoming events” on page 169

### Page Container widget outgoing events:

The Page Container widget uses events to communicate with other widgets.

### Open page event

Description	The user opened the page.
Event ID	icm.OpenPage
Type	Broadcast

---

Payload

**pageClassName**

A string that contains the name of the page class module.

**pageType**

A string that contains the one of the following values to indicate the type of page to be opened:

**CASE** The page is a Case Details page.

**CASE\_NEW**

The page is an Add Case page.

**CASE\_SPLIT**

The page is a Split Case page.

**STATIC**

The page is a Solution page.

**STEP** The page is a Work Details page.

**STEP\_LAUNCH'**

The page is an Add Task page.

**isLazy**

A Boolean value that is set to True to indicate that the page is to be lazy loaded. That is, the page is not selected immediately and is loaded only when it is selected manually.

**subject**

A model object or a simple string that represents the page subject. For example, you might use an `icm.model.CaseEditable` object for a Case Detail page or a string containing the Universally Unique Identifier (UUID) for a Work page.

**pageContext**

An object that represents the context that is to be added as a property of the page and each page widget.

**crossPageEventName**

A string that contains the name of the event that is to be broadcast in the page.

**crossPageEventPayload**

An object that represents the payload of the event that is to be broadcast in the page.

---

**Page activated event**

---

Description	The user selected the page.
Event ID	<code>icm.PageActivated</code>
Type	Broadcast
Payload	<code>null</code>

---

**Page closed event**

---

Description	The user closed the page.
Event ID	<code>icm.PageClosing</code>
Type	Broadcast

---

Payload	null
---------	------

### Page deactivated event

Description	The user selected a different page, which changed focus from the current page.
Event ID	icm.PageDeactivated
Type	Broadcast
Payload	null

### Page opened event

Description	The user opened the page.
Event ID	icm.OpenPage
Type	Broadcast
Payload	null

### Send case information event

Description	The user opened a case.
Event ID	icm.SendCaseInfo
Type	Broadcast
Payload	<p><b>caseEditable</b> An icm.model.CaseEditable object that represents the case that is to be displayed</p> <p><b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.</p>

### Send new case information event

Description	The user clicked the <b>Add Case</b> button.
Event ID	icm.SendNewCaseInfo
Type	Broadcast
Payload	<p><b>caseEditable</b> An icm.model.CaseEditable object that represents the case that is to be created</p> <p><b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.</p>

### Send new task information event

Description	The user clicked the <b>Add Task</b> button.
Event ID	icm.SendNewTaskInfo
Type	Broadcast

---

Payload	<p><b>taskEditable</b> An <code>icm.model.TaskEditable</code> object that represents the task that is to be created.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>
---------	---

---

### Send split case information event

---

Description	The user clicked the <b>Split Case</b> button.
Event ID	<code>icm.SendSplitCaseInfo</code>
Type	Broadcast
Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be created</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>

---

### Send work item event

---

Description	Send the work item.
Event ID	<code>icm.SendWorkItem</code>
Type	Broadcast
Payload	<p><b>workItemEditable</b> An <code>icm.model.WorkItemEditable</code> object that represents the work item that is to be displayed and updated.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>

---




### Widgets loaded event

---

Description	Widgets in the page are loaded and ready to work.
Event ID	<code>icm.WidgetLoaded</code>
Type	Broadcast
Payload	<p><b>widgetLoadTime</b> An <code>icm.context.widgetLoadTime</code> object that represents the time it takes for the widgets to load.</p>

---

#### Related concepts:

-  [Class `icm.model.CaseEditable`](#)
-  [Class `icm.model.TaskEditable`](#)
-  [Class `icm.base.Coordination`](#)

## Page Container widget incoming events:

The Page Container widget uses events to communicate with other widgets.

### Add case event

Description	Open an Add Case page so the user can add a case.
Event ID	icm.AddCase
Payload	<b>caseType</b> An icm.model.CaseType object that represents the type of case to be added. <b>caseEditable</b> An icm.model.CaseEditable object that represents the case that is to be added. <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

### Add task event

Description	Open an Add Task page so that the user can add a task.
Event ID	icm.AddTask
Payload	<b>taskEditable</b> An icm.model.TaskEditable object that represents the task that is to be added. <b>caseEditable</b> An icm.model.CaseEditable object that represents the case to which the task to be added <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

### Close page event

Description	Close the current page.
Event ID	icm.ClosePage
Payload	Empty

### Collapse region event

Description	Collapse a region in the page.
Event ID	icm.CollapseRegion
Payload	<b>region</b> A string that identifies the region of the page to be collapsed. The valid values are leading, trailing, top, and bottom. You can specify center as the region, but the center region cannot be collapsed.

### Mark or clear page as changed event

Description	Set or clear the flag that indicates whether the page contains unsaved changes.
Event ID	icm.SetDirtyState
Payload	<b>reference</b> A reference that identifies the entity that invoked this event.

### Open case event

Description	Open the specified case in a Case Details page.
Event ID	icm.OpenCase
Payload	<b>caseEditable</b> An icm.model.CaseEditable object that represents the case that is to be opened. <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.

### Open page event

Description	Open the page that is specified in the event payload.
Event ID	icm.OpenPage



---

Payload

**pageClassName**

A string that contains the name of the page class module.

**pageType**

A string that contains the one of the following values to indicate the type of page to be opened:

**CASE** The page is a Case Details page.

**CASE\_NEW**

The page is an Add Case page.

**CASE\_SPLIT**

The page is a Split Case page.

**STATIC**

The page is a Solution page.

**STEP** The page is a Work Details page.

**STEP\_LAUNCH'**

The page is an Add Task page.

**isLazy**

A Boolean value that is set to True to indicate that the page is to be lazy loaded. That is, the page is not selected immediately and is loaded only when it is selected manually.

**subject**

A model object or a simple string that represents the case or work item that is to be opened in the page. For example, you might use an `icm.model.CaseEditable` object for a Case Detail page or a string containing the Universally Unique Identifier (UUID) for a Work page.

**pageContext**

An object that represents the context that is to be added as a property of the page and each page widget.

**crossPageEventName**

A string that contains the name of the event that is to be broadcast in the page.

**crossPageEventPayload**

An object that represents the payload of the event that is to be broadcast in the page.

---

## Open work item event

---

Description	Open the specified work item in a Work Details page.
-------------	--

---

Event ID	<code>icm.OpenWorkItem</code>
----------	-------------------------------

Payload

**workItem**

An `icm.model.WorkItemEditable` object that represents the work item that is to be opened.

**coordination**

An `icm.util.Coordination` object that is used internally by the widgets in the same page.

**UIState**

A `dojo.Stateful` object that represents the state of the selected work item.

---

## Restore region event

Description	Restore a region in the page to its original size.
Event ID	icm.RestoreRegion
Payload	<b>region</b> A string that identifies the region of the page to be restored. The valid values are leading, trailing, top, and bottom.

## Split case event

Description	Open a Split Case page so that the user can create a case from an existing case.
Event ID	icm.SplitCase
Payload	<b>caseType</b> An <code>icm.model.CaseType</code> object that represents the type of case to be created from the existing case. <b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be split. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

### Related concepts:

- [↗](#) Class `icm.model.CaseEditable`
- [↗](#) Class `icm.model.CaseType`
- [↗](#) Class `icm.model.TaskEditable`
- [↗](#) Class `icm.model.WorkItemEditable`
- [↗](#) `icm.base.Coordination`

## Process History widget events

The Process History widget handles incoming events to display the history for tasks and work items.

“Process History widget incoming events”

### Process History widget incoming events:

The Process History widget provides incoming events to handle the data that is received from other widgets.

## Clear Content event

Description	Clear the content from the Process History widget.
Event ID	icm.ClearContent
Payload	null

### Open work item event

Description	Display the process history for the work item that is contained in the event payload.
Event ID	icm.SendWorkItem
Payload	<b>workItem</b> An icm.model.WorkItem object that represents the work item for which process history is to be displayed.

### Receive task information event

Description	Display the process history for the task that is contained in the event payload.
Event ID	icm.TaskSelect
Payload	<b>task</b> An icm.model.Task object that represents the task for which process history is to be displayed.  Alternatively, the payload can be null.

#### Related concepts:

 [Class icm.model.Task](#)

 [Class icm.model.WorkItem](#)

### Properties widget events

The Properties widget publishes and handles events to display case and work item properties for users to view and edit.

For example, the Properties widget handles an incoming event to display the properties for a case type so that the user can create a case of that type.

“Properties widget outgoing events”

“Properties widget incoming events” on page 182

#### Properties widget outgoing events:

The Properties widget provides outgoing events that are broadcast and wired to the other widgets on the page. The data that is included in the payload of a broadcast event is processed by any widget that has a corresponding handled event. The data that is included in the payload of a wired event is processed by the handled event in the target widget to which the published widget is wired.

### Cell added event

Description	A cell was added to a property table.
Event ID	icm.CellAdded
Type	Broadcast

---

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was added.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the new cell.</p> <p><b>row</b> The row index of the new cell.</p> <p><b>error</b> A single value string or array that represents the error message of the new cell.</p> <p><b>value</b> The value of the new cell.</p>
---------	--

---

### Cell changed event

Description	The value of a property table cell was changed. This event is published when the value or error message of a cell is modified.
Event ID	<code>icm.CellChanged</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell value was changed.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the updated cell.</p> <p><b>row</b> The row index of the updated cell.</p> <p><b>error</b> A single value string or array that represents the error message of the updated cell.</p> <p><b>value</b> The updated value of the cell.</p>

---

### Cell clicked event

Description	A cell in a property table was clicked.
Event ID	<code>icm.CellClicked</code>
Type	Broadcast

---

---

Payload

**propertyTable**

A `pvr.widget.PropertyTable` object that represents the property table in which a cell was clicked.

**property**

A `pvr.widget.Property` object that represents the property that is associated with the clicked cell.

**row**

The row index of the clicked cell.

**value**

The value of the clicked cell.

**error**

A single value string or array that represents the error message of the clicked cell.

---

### Cell double clicked event

---

Description	A cell was double-clicked within a property table in the view.
Event ID	<code>icm.CellDoubleClicked</code>
Type	Broadcast

---

Payload

**propertyTable**

A `pvr.widget.PropertyTable` object that represents the property table in which a cell was double-clicked.

**property**

A `pvr.widget.Property` object that represents the property that is associated with the double-clicked cell.

**row**

The row index of the double-clicked cell.

**value**

The value of the double-clicked cell.

**error**

A single value string or array that represents the error message of the double-clicked cell.

---

### Cell inserted event

---

Description	A cell was inserted in a property table.
Event ID	<code>icm.CellInserted</code>
Type	Broadcast

---

---

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was inserted.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the inserted cell.</p> <p><b>row</b> The row index of the inserted cell.</p> <p><b>error</b> A single value string or array that represents the error message of the inserted cell.</p> <p><b>value</b> The value of the inserted cell.</p>
---------	---

---

### Cell moved event

Description	A cell in a property table was moved.
Event ID	<code>icm.CellMoved</code>
Type	Broadcast

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was moved.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the moved cell.</p> <p><b>row</b> The new row index of the moved cell.</p> <p><b>error</b> A single value string or array that represents the error message of the moved cell.</p> <p><b>value</b> The value of the moved cell.</p>
---------	--

---

### Cell removed event

Description	A cell was removed from a property table.
Event ID	<code>icm.CellRemoved</code>
Type	Broadcast

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was removed.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the removed cell.</p> <p><b>row</b> The row index of the removed cell.</p> <p><b>error</b> A single value string or array that represents the error message of the removed cell.</p> <p><b>value</b> The value of the removed cell.</p>
---------	--

### Cell updated event

Description	A property table cell was updated. This event is published when a cell is modified as a result of an Add, Insert, Move, or Remove operation.
Event ID	<code>icm.CellUpdated</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was updated.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the updated cell.</p> <p><b>row</b> The row index of the updated cell.</p> <p><b>error</b> A single value string or array that represents the error message of the updated cell.</p> <p><b>value</b> The updated value of the cell.</p>

### Field blurred event

Description	The field lost focus for user input.
Event ID	<code>icm.FieldBlurred</code>
Type	Broadcast

---

Payload	<p><b>id</b> A string that contains the identifier of the property with which the field is associated.</p> <p>The <b>collectionId</b> parameter identifies the type of properties that were updated. This parameter will have one of the following values:</p> <p><b>F_CaseFolder</b> Indicates that this is a case property.</p> <p><b>F_CaseTask</b> Indicates that this is a task property.</p> <p><b>F_WorkflowField</b> Indicates that this is a work group or workflow data field.</p> <p>The data type of the value corresponds to the data type of the property, or is null. For properties that support multiple values, the value is an array of the corresponding data type.</p>
---------	---

---

### Field focused event

---

Description	The field got focus for user input.
Event ID	icm.FieldFocused
Type	Broadcast
Payload	<p><b>id</b> A string that contains the identifier of the property with which the field is associated.</p> <p>The <b>collectionId</b> parameter identifies the type of properties that were updated. This parameter will have one of the following values:</p> <p><b>F_CaseFolder</b> Indicates that this is a case property.</p> <p><b>F_CaseTask</b> Indicates that this is a task property.</p> <p><b>F_WorkflowField</b> Indicates that this is a work group or workflow data field.</p> <p>The data type of the value corresponds to the data type of the property, or is null. For properties that support multiple values, the value is an array of the corresponding data type.</p>

---

### Field updated event

---

Description	The value of a field was updated.
Event ID	icm.FieldUpdated
Type	Broadcast

---



---

## Payload

### **change**

A JSON object that contains the unique identifier, collection identifier, and value for the property that was modified.

The structure is:

```
change: [
  {
    id: prop1,
    collectionId: collectionId1,
    value: value1
  }
]
```

The **id** parameter is a string that contains the identifier of the property with which the field is associated.

The **collectionId** parameter identifies the type of properties that were updated. This parameter will have one of the following values:

#### **F\_CaseFolder**

Indicates that this is a case property.

#### **F\_CaseTask**

Indicates that this is a task property.

#### **F\_WorkflowField**

Indicates that this is a work group or workflow data field.

The data type of the value corresponds to the data type of the property, or is null. For properties that support multiple values, the value is an array of the corresponding data type.

---

## Property updated event

---

Description	The value of a property was updated.
Event ID	icm.PropertyUpdated
Type	Broadcast

---

### Payload

#### **property**

A `pvr.widget.Property` object that represents the property whose value was updated.

#### **value**

The updated value of the property. The data type of the value corresponds to the data type of the property, or is null. For properties that support multiple values, the value is an array of the corresponding data type.

#### **error**

A single value string or array that represents the error message.

---

## Row added event

---

Description	A row was added in a property table.
Event ID	icm.RowAdded
Type	Broadcast

---

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was added.</p> <p><b>row</b> The index of the new row.</p>
---------	---

### Row changed event

Description	A row in a property table was changed. This event is published when the value or error message of one of the cells in a row is modified.
Event ID	<code>icm.RowChanged</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was changed.</p> <p><b>row</b> The index of the changed row.</p>

### Row clicked event

Description	A row in a property table was clicked.
Event ID	<code>icm.RowClicked</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was clicked.</p> <p><b>row</b> The index of the clicked row.</p>

### Row double clicked event

Description	A row was double-clicked within a property table in the view.
Event ID	<code>icm.RowDoubleClicked</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was double-clicked.</p> <p><b>row</b> The index of the double-clicked row.</p>

### Row inserted event

Description	A row was inserted in a property table.
Event ID	<code>icm.RowInserted</code>
Type	Broadcast

---

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was inserted.</p> <p><b>row</b> The index of the inserted row.</p>
---------	---

---

### Row moved event

---

Description	A row in a property table was moved.
Event ID	<code>icm.RowMoved</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was moved.</p> <p><b>row</b> The new index of the moved row.</p>

---

### Row removed event

---

Description	A row was removed from a property table.
Event ID	<code>icm.RowRemoved</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was removed.</p> <p><b>row</b> The index of the removed row.</p>

---

### Row selected event

---

Description	The row selection changed in a property table in the view.
Event ID	<code>icm.RowSelected</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was selected.</p> <p><b>row</b> The index of the selected row.</p>

---

### Row updated event

---

Description	A row in a property table was updated. This event is published when a row is modified as a result of an Add, Insert, Move, or Remove operation.
Event ID	<code>icm.RowUpdated</code>
Type	Broadcast

---

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was updated.</p> <p><b>row</b> The index of the updated row.</p>
---------	---

### Properties widget incoming events:

The Properties widget provides the following incoming events to handle the data that is received from other widgets.

#### Update fields event

Description	Update the fields with the values that are contained in the event payload.
Event ID	<code>icm.UpdateFields</code>
Payload	<p><b>changes</b> A JSON object that contains the unique identifier, collection identifier, and value for the property to be modified. The structure is:</p> <pre>changes:[   {     id: prop1,     collectionId: collectionId1,     value: value1   },   {     id: prop2,     collectionId: collectionId2,     value: value2   } ]</pre> <p>The <b>collectionId</b> parameter identifies the type of properties that are being updated. Use the following values for this parameter:</p> <p><b>F_CaseFolder</b> Use this value for case properties.</p> <p><b>F_CaseTask</b> Use this value for task properties.</p> <p><b>F_WorkflowField</b> Use this value for work group or workflow data fields.</p> <p>The data type of each value must correspond to the data type of the property, or must be null. For properties that support multiple values, the value is an array of the corresponding data type. If the value is incompatible, the value is coerced if possible or a default value is applied. A warning is also logged.</p>

#### Update properties event

Description	Update the properties with the values that are contained in the event payload.
Event ID	<code>icm.UpdateProperties</code>

---

## Payload

### changes

A JSON object that contains the `pvr.widget.Property` object, value, error, and row for the property to be updated. The structure is:

```
changes: [  
  {  
    property: prop1,  
    value: value1,  
    [error]: error1,  
    [row]: row1  
  },  
  {  
    property: prop2,  
    value: value2,  
    [error]: error2,  
    [row]: row2  
  }  
]
```

The data type of each value must correspond to the data type of the property, or must be null. For properties that support multiple values, the value is an array of the corresponding data type. If the value is incompatible, the value is coerced if possible or a default value is applied. A warning is also logged.

---

## Clear content event

---

Description	Clear the content in the Case List widget.
Event ID	<code>icm.ClearContent</code>
Payload	<code>null</code>

---

## Send case information event

---

Description	Display the properties view for the specified case. Replace the existing view if one exists.
Event ID	<code>icm.SendCaseInfo</code>
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be displayed and updated. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

---

## Send new case information event

---

Description	Display the properties for the new case that the user is creating.
Event ID	<code>icm.SendNewCaseInfo</code>

---

---

Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be created.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>
---------	---

---

### Send new task information event

---

Description	Display the properties for the new task that the user is creating.
Event ID	<code>icm.SendNewTaskInfo</code>
Payload	<p><b>taskEditable</b> An <code>icm.model.TaskEditable</code> object that represents the task that is to be created. The editable properties are contained in the <code>icm.model.LaunchStep</code> object that is associated with the <code>icm.model.TaskEditable</code> object. You can obtain the editable properties by calling the <code>taskEditable.getLaunchStep()</code> method.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>

---





### Send work item event

---

Description	Display the properties for the work item that is contained in the event payload.
Event ID	<code>icm.SendWorkItem</code>
Payload	<p><b>workItemEditable</b> An <code>icm.model.WorkItemEditable</code> object that represents the work item that is to be displayed and updated.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>

---

#### Related concepts:

-  [Class `icm.model.CaseEditable`](#)
-  [Class `icm.model.TaskEditable`](#)
-  [Class `icm.model.WorkItemEditable`](#)
-  [Class `icm.base.Coordination`](#)

### Script Adapter widget events

The Script Adapter widget uses events to insert logic to transform or validate event data between the widgets on a page.

“Script Adapter widget outgoing events” on page 185

“Script Adapter widget incoming events” on page 185

### Script Adapter widget outgoing events:

The Script Adapter widget provides an outgoing event that is broadcast to the other widgets on the page. The data that is included in the payload of this broadcast event is processed by any widget that has a corresponding handled event.

#### Send event payload event

Description	The data in the payload that was received in the incoming event was converted based on the script that is defined for the Script Adapter widget.
Event ID	<code>icm.SendEventPayload</code>
Type	Wired
Payload	The payload is determined by the script that is configured for the Script Adapter widget.

### Script Adapter widget incoming events:

The Script Adapter widget provides an incoming event to handle the data that is received from other widgets.

#### Receive event payload event

Description	Convert the data in the payload of the incoming event according to the script that is provided for the Script Adapter widget.
Event ID	<code>icm.ReceiveEvent</code>
Payload	The payload is determined by the script that is configured for the Script Adapter widget.

### Search widget events

The Search widget uses events to enable users to search for cases in a solution.

For example, the Search widget publishes an outgoing event that contains the search criteria that a user specified.

“Search widget outgoing events”

“Search widget incoming events” on page 186

#### Search widget outgoing events:

The Search widget provides an outgoing event that is broadcast to the other widgets on the page. The data that is included in the payload of this broadcast event is processed by any widget that has a corresponding handled event.

#### Search cases event

Description	The user started a search for cases.
Event ID	<code>icm.SearchCases</code>
Type	Broadcast

---

## Payload

### **caseType**

The symbolic name of the case type that is being searched. If the search is across a solution, the value of this parameter is "".

### **searchTemplate**

An `ecm.model.SearchTemplate` object that represents the template that is being used for the search.

### **caseTypeTitles**

A JSON object that identifies the titles of the type of cases being searched for. The structure of this object is as follows:

```
{
  "caseType1SymbolicName": "title1SymbolicName",
  "caseType2SymbolicName": "title2SymbolicName",
  "caseType3SymbolicName": "title3SymbolicName",
}
```

### **detailsViewProperties**

An array of JSON objects that identify the properties that are to be displayed in the details view of the Case List widget. Each property element in the array has the following structure:

```
{display name, orderable, symbolic name, type}
```

### **magazineViewProperties**

An array of JSON objects that identify the properties that are to be displayed in the magazine view of the Case List widget. Each property element in the array has the following structure:


```
{display name, orderable, symbolic name, type}
```

### **ceQuery**

A user-provided Content Engine query statement that is used to search cases.

---

## **Related concepts:**

 [Class `ecm.model.SearchTemplate`](#)

## **Search widget incoming events:**

The Search widget provides an incoming event to handle the data that is received from other widgets.

### **Clear content event**

Description	Clear the content in the Search widget.
Event ID	<code>icm.ClearContent</code>
Payload	<code>null</code>

---

## **Select Case Documents widget events**

The Select Case Documents widget handles incoming events to display documents from one case so that the user can add them to a new case..

“Select Case Documents widget incoming events” on page 187



## Select Case Documents widget incoming events:

The Select Case Documents widget provides the incoming events to handle the data that is received from other widgets.

### Clear content event

Description	Clear the content in the Select Case Document widget. The original case documents are not displayed and the list of selected case documents is cleared.
Event ID	<code>icm.ClearContent</code>
Payload	<code>null</code>

### Clear selected documents event

Description	Clear the documents that the user has selected in the Select Case Documents widget.
Event ID	<code>icm.ClearSelectedDoc</code>
Payload	<code>null</code>

### Send case information event

Description	Display the documents from the original case that the user can add to the new case.
Event ID	<code>icm.SendSplitCaseInfo</code>
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be created. <b>caseType</b> An <code>icm.model.CaseType</code> object that represents the type of case that is to be created. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by widgets in the same page for cooperation.

### Related concepts:

- [↗](#) Class `icm.model.CaseEditable`
- [↗](#) Class `icm.model.CaseType`
- [↗](#) `icm.base.Coordination`

## Split Case Properties widget events

The Split Case Properties widget handles incoming events to enable a user to create a case by reusing property values from an existing case.

“Split Case Properties widget outgoing events” on page 188

“Split Case Properties widget incoming events” on page 195

## Split Case Properties widget outgoing events:

The Split Case Properties widget provides outgoing events that are broadcast to the other widgets on the page. The data that is included in the payload of a broadcast event is processed by any widget that has a corresponding handled event.

### Cell added event

Description	A new cell was added to a property table in the view.
Event ID	icm.CellAdded
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was added.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the new cell.</p> <p><b>row</b> The row index of the new cell.</p> <p><b>error</b> A single value string or array that represents the error message of the new cell.</p> <p><b>value</b> The value of the new cell.</p>

### Cell changed event

Description	The value of a property table cell in the view was changed.
Event ID	icm.CellChanged
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell value was changed.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the updated cell.</p> <p><b>row</b> The row index of the updated cell.</p> <p><b>error</b> A single value string or array that represents the error message of the updated cell.</p> <p><b>value</b> The updated value of the cell.</p>

### Cell clicked event

Description	A cell was clicked within a property table in the view.
Event ID	icm.CellClicked

Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was clicked.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the clicked cell.</p> <p><b>row</b> The row index of the clicked cell.</p> <p><b>value</b> The value of the clicked cell.</p> <p><b>error</b> A single value string or array that represents the error message of the clicked cell.</p>

### Cell double clicked event

Description	A cell was double clicked within a property table in the view.
Event ID	<code>icm.CellDoubleClicked</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was double-clicked.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the double-clicked cell.</p> <p><b>row</b> The row index of the double-clicked cell.</p> <p><b>value</b> The value of the double-clicked cell.</p> <p><b>error</b> A single value string or array that represents the error message of the double-clicked cell.</p>

### Cell inserted event

Description	A new cell was inserted into a property table in the view.
Event ID	<code>icm.CellInserted</code>
Type	Broadcast

---

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was inserted.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the inserted cell.</p> <p><b>row</b> The row index of the inserted cell.</p> <p><b>error</b> A single value string or array that represents the error message of the inserted cell.</p> <p><b>value</b> The value of the inserted cell.</p>
---------	---

---

### Cell moved event

---

Description	A cell was moved within a property table in the view.
Event ID	<code>icm.CellMoved</code>
Type	Broadcast

---

Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a cell was moved.</p> <p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property that is associated with the moved cell.</p> <p><b>row</b> The new row index of the moved cell.</p> <p><b>error</b> A single value string or array that represents the error message of the moved cell.</p> <p><b>value</b> The value of the moved cell.</p>
---------	--

---

### Cell removed event

---

Description	A cell was removed from a property table in the view.
Event ID	<code>icm.CellRemoved</code>
Type	Broadcast

---

---

**Payload****propertyTable**

A `pvr.widget.PropertyTable` object that represents the property table in which a cell was removed.

**property**

A `pvr.widget.Property` object that represents the property that is associated with the removed cell.

**row**

The row index of the removed cell.

**error**

A single value string or array that represents the error message of the removed cell.

**value**

The value of the removed cell.

---

**Cell updated event**

---

Description	The cell in the property table in the view was updated.
Event ID	<code>icm.CellUpdated</code>
Type	Broadcast

---

**Payload****propertyTable**

A `pvr.widget.PropertyTable` object that represents the property table in which a cell was updated.

**property**

A `pvr.widget.Property` object that represents the property that is associated with the updated cell.

**row**

The row index of the updated cell.

**error**

A single value string or array that represents the error message of the updated cell.

**value**

The updated value of the cell.

---

**Field updated event**

---

Description	The value of a field was updated.
Event ID	<code>icm.FieldUpdated</code>
Type	Broadcast

---

---

## Payload

### changes

A JSON object that contains the unique identifier and value for the property that was modified. The structure is:

```
change: [  
  {  
    id: prop1,  
    value: value1  
  }  
]
```

The **collectionId** value identifies the type of properties that were updated. This parameter will have one of the following values.

#### F\_CaseFolder

Indicates that this is a case property.

#### F\_CaseTask

Indicates that this is a task property.

#### F\_WorkflowField

Indicates that this is a work group or workflow data field.

The data type of the value corresponds to the data type of the property, or is null. For properties that support multiple values, the value is an array of the corresponding data type.

---

## Field focused event

---

Description	The field got focus for user input.
Event ID	icm.FieldFocused
Type	Broadcast
Payload	<b>id</b> A string that contains the identifier of the property with which the field is associated.

---

## Field blurred event

---

Description	The field lost focus for user input.
Event ID	icm.FieldBlurred
Type	Broadcast
Payload	<b>id</b> A string that contains the identifier of the property with which the field is associated.

---

## Property updated event

---

Description	The value of a property in the view was updated.
Event ID	icm.PropertyUpdated
Type	Broadcast

---

---

Payload	<p><b>property</b> A <code>pvr.widget.Property</code> object that represents the property whose value was updated.</p> <p><b>value</b> The updated value of the property. The data type of the value corresponds to the data type of the property, or is null. For properties that support multiple values, the value is an array of the corresponding data type.</p> <p><b>error</b> A single value string or array that represents the error message.</p>
---------	---

---

### Row added event

---

Description	A new row was added to a property table in the view.
Event ID	<code>icm.RowAdded</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was added.</p> <p><b>row</b> The index of the new row.</p>

---

### Row changed event

---

Description	A row was changed within a property table in the view.
Event ID	<code>icm.RowChanged</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was changed.</p> <p><b>row</b> The index of the changed row.</p>

---

### Row clicked event

---

Description	A row was clicked within a property table in the view.
Event ID	<code>icm.RowClicked</code>
Type	Broadcast
Payload	<p><b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was clicked.</p> <p><b>row</b> The index of the clicked row.</p>

---

### Row double clicked event

Description	A row was double clicked within a property table in the view.
Event ID	<code>icm.RowDoubleClicked</code>
Type	Broadcast
Payload	<b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was double-clicked.  <b>row</b> The index of the double-clicked row.

### Row inserted event

Description	A new row was inserted into a property table in the view.
Event ID	<code>icm.RowInserted</code>
Type	Broadcast
Payload	<b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was inserted.  <b>row</b> The index of the inserted row.

### Row moved event

Description	A row was moved within a property table in the view.
Event ID	<code>icm.RowMoved</code>
Type	Broadcast
Payload	<b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was moved.  <b>row</b> The new index of the moved row.

### Row removed event

Description	A row was removed from a property table in the view.
Event ID	<code>icm.RowRemoved</code>
Type	Broadcast
Payload	<b>propertyTable</b> A <code>pvr.widget.PropertyTable</code> object that represents the property table in which a row was removed.  <b>row</b> The index of the removed row.



### Row selected event

Description	The row selection changed in a property table in the view.
Event ID	icm.RowSelected
Type	Broadcast
Payload	<b>propertyTable</b> A pvr.widget.PropertyTable object that represents the property table in which a row was selected.  <b>row</b> The index of the selected row.

### Row updated event

Description	A row was updated within a property table in the view.
Event ID	icm.RowUpdated
Type	Broadcast
Payload	<b>propertyTable</b> A pvr.widget.PropertyTable object that represents the property table in which a row was updated.  <b>row</b> The index of the updated row.

### Split Case Properties widget incoming events:

The Split Case Properties widget provides incoming events to handle the data that is received from other widgets.

### Clear content event

Description	Clear the content in the Case List widget.
Event ID	icm.ClearContent
Payload	null

### Split Case Properties event

Description	Display the properties for creating a case and reuse corresponding property values from the existing case.
Event ID	icm.SendSplitCaseInfo
Payload	<b>caseEditable</b> An icm.model.CaseEditable object that represents the case that is to be created.  <b>coordination</b> An icm.util.Coordination object that is used internally by the widgets in the same page.


## Update fields event

Description	Update the fields with the values that are contained in the event payload.
Event ID	icm.UpdateFields
Payload	<p><b>changes</b></p> <p>A JSON object that contains the unique identifier and value for the property that was modified. The structure is:</p> <pre>change:[   {     id: prop1,     value: value1   }   {     id: prop1,     value: value1   } ]</pre> <p>The data type of each value must correspond to the data type of the property, or must be null. For properties that support multiple values, the value is an array of the corresponding data type. If the value is incompatible, the value is coerced if possible or a default value is applied. A warning is also logged.</p>

## Update properties event

Description	Update the properties in the view with the values that are contained in the event payload.
Event ID	icm.UpdateProperties
Payload	<p><b>changes</b></p> <p>A JSON object that contains the pvr.widget.Property object, value, error, and row for the property to be updated. The structure is:</p> <pre>changes:[   {     property: prop1,     value: value1,     [error]: error1,     [row]: row1   },   {     property: prop2,     value: value2,     [error]: error2,     [row]: row2   } ]</pre> <p>The data type of each value must correspond to the data type of the property, or must be null. For properties that support multiple values, the value is an array of the corresponding data type. If the value is incompatible, the value is coerced if possible or a default value is applied. A warning is also logged.</p>

### Related concepts:

 Class `icm.model.CaseEditable`

## Timeline Visualizer widget events

The Timeline Visualizer widget publishes outgoing events to display the extended history for a case.

For example, the Timeline Visualizer widget publishes the Select task event when the user selects a task in the widget.

“Timeline Visualizer widget outgoing events”

“Timeline Visualizer widget incoming events” on page 198

### Timeline Visualizer widget outgoing events:

The Timeline Visualizer widget provides the following outgoing events that are broadcast to the other widgets on the page. The data that is included in the payload of a broadcast event is processed by any widget that has a corresponding handled event.

#### Select case event

Description	The user selected the case in the Timeline Visualizer widget.
Event ID	<code>icm.SelectTimelineCase</code>
Type	Broadcast
Payload	<b>timeline</b> An <code>icm.model.Timeline</code> object that represents the Timeline API that is used to retrieve case history data. <b>case</b> An <code>icm.model.Case</code> object that represents the case that is associated with the timeline that the user selected <b>timelineOverview</b> An <code>icm.model.TimelineOverview</code> object that contains general case history data, such as the case identifier, name, and status; and events density data.

#### Select events event

Description	The user selected one or more events in the Timeline Visualizer widget.
Event ID	<code>icm.SelectTimelineEvents</code>
Type	Broadcast
Payload	<b>timeline</b> An <code>icm.model.Timeline</code> object that represents the timeline that is used to retrieve case history data. <b>case</b> An <code>icm.model.Case</code> object that represents the case that is associated with the events that the user selected <b>timelineEvents</b> An array of <code>icm.model.TimelineEvent</code> objects that represents the events that are selected by the user.

### Select task event

Description	The user selected a task in the Timeline Visualizer widget.
Event ID	icm.SelectTimelineTask
Type	Broadcast
Payload	<b>timeline</b> An <code>icm.model.Timeline</code> object that represents the timeline that is used to retrieve case history data. <b>case</b> An <code>icm.model.Case</code> object that represents the case that is associated with the task that the user selected <b>timelineTask</b> An <code>icm.model.TimelineTask</code> object that represents the task that is selected by the user.

### Select work item event

Description	The user selected a work item in the Timeline Visualizer widget.
Event ID	icm.SelectTimelineWorkItem
Type	Broadcast
Payload	<b>timeline</b> An <code>icm.model.Timeline</code> object that represents the timeline that is used to retrieve case history data. <b>case</b> An <code>icm.model.Case</code> object that represents the case that is associated with the work item that the user selected <b>timelineWorkItem</b> An <code>icm.model.TimelineWorkItem</code> object that represents the work item that is selected by the user.

#### Related concepts:

- [icm.model.Timeline](#)
- [icm.model.TimelineEvent](#)
- [icm.model.TimelineOverview](#)
- [icm.model.TimelineTask](#)

#### Timeline Visualizer widget incoming events:

The Timeline Visualizer widget provides the following incoming events to handle the data that is received from other widgets.

### Clear content event

Description	Clear the content of the Timeline Visualizer widget.
Event ID	icm.ClearContent
Payload	null

## Open work item event

Description	Display the extended history in the Timeline Visualizer for the case that is contained in the event payload. This event is used when the Timeline Visualizer widget is included on a Work Details page.
Event ID	<code>icm.SendWorkItem</code>
Payload	<b>workItemEditable</b> An <code>icm.model.WorkItemEditable</code> object that represents the work item for which the case history is to be displayed. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.


## Select case event

Description	Display the extended history in the Timeline Visualizer for the case that is contained in the event payload. This event is used when the Timeline Visualizer widget is included on a Case Details page.
Event ID	<code>icm.SelectCase</code>
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case for which the extended history is to be displayed.

## Send case information event

Description	Display the extended history in the Timeline Visualizer for the case that is contained in the event payload. This event is used when the Timeline Visualizer widget is included on a Case Details page.
Event ID	<code>icm.SendCaseInfo</code>
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case for which the extended history is to be displayed. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.

### Related concepts:

 [Class `icm.model.CaseEditable`](#)

 [`icm.base.Coordination`](#)

## To-Do List widget events

The To-Do List widget uses incoming events to display a list of items that require handling by the user.

“To-Do List widget incoming events” on page 200

## To-Do List widget incoming events:

The To-Do List widget provides incoming events to handle the data that is received from other widgets.

### Create Quick Task event

Description	Create a quick task and display in the To-Do List widget.
Event ID	icm.QuickTaskCreate
Payload	<b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case for which the quick task is to be created. <b>quickTaskName</b> A string that contains the name of the quick task. <b>description</b> A string that contains a description for the quick task. <b>dueDate</b> A string that contains the date that the quick task must be complete. The date is in the format MM/DD/YYYY.

### Receive work item event

Description	Displays the to-do list associated with the case containing the work item that is contained in the event payload.
Event ID	icm.SendWorkItem
Payload	<b>workItemEditable</b> An <code>icm.model.WorkItemEditable</code> object that represents the work item to be opened. <b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page. <b>UIState</b> A Dojo Stateful object that is used internally by the widgets. This object can contain the following properties: <b>GetNext</b> This property determines whether the next work item is to be opened automatically. <b>workitemReadonly</b> This property determines whether the work item is to be opened in view mode. <b>GetNextCfg</b> This property determines the configuration of the Open Next Work Item action.

### Receive case event

Description	Displays the to-do list associated with the case that is contained in the event payload.
Event ID	icm.SendCaseInfo

---

Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be displayed.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>
---------	---

---

### Select case event

---

Description	Displays the to-do list associated with the case that is contained in the event payload.
Event ID	<code>icm.SelectCase</code>
Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be displayed.</p>

---

### To-do task added event

---

Description	Adds the to-do task to display in the To-Do List widget.
Event ID	<code>icm.ToDoTaskAdded</code>
Payload	<p><b>ToDoTaskEditable</b> A <code>TaskEditable</code> object that represents the added to-do task.</p>

---

### Refresh to-do list event

---

Description	Refreshes the list of to-do tasks in the To-Do List widget.
Event ID	<code>icm.RefreshToDoTaskList</code>
Payload (optional)	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case that is to be displayed.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>

---

### Optional payloads

The following payloads are optional for all To-Do List widget incoming events:

#### **propertyFilter**

Defines the conditions for tasks to be returned.

#### **propertyList**

Contains the properties that were returned by a call to the `icm.model.Case.searchTasks` method.

#### **resultDisplay**

An optional JavaScript object that defines the sort display attributes of the returned results.

**includeHidden**

Set this parameter to true to include hidden tasks in the results. Otherwise, hidden tasks are filtered out.

**gridStructure**

Defines which columns are shown, the widths of the columns, the display name of each column header, and which columns can be sorted.

**Toolbar widget events**

No events are defined specifically for the Toolbar widget. However, the Toolbar widget does publish events if certain actions are configured for the widget. For example, if the Add Case action is configured for the Toolbar widget, the widget publishes the Open new case page event for that action..

**Viewer widget events**

The Viewer widget handles incoming events to display case documents and work item attachments to users.

“Viewer widget incoming events”

**Viewer widget incoming events:**

The Viewer widget provides the incoming events to handle the data that is received from other widgets.

**Clear content event**

Description	Clear the content of Viewer widget and the cached identifiers, and then close all tabs.
Event ID	icm.ClearContent
Payload	null

**Open document event**

Description	Open the document that is contained in the event payload in the Viewer widget.
Event ID	icm.OpenDocument
Payload	<p><b>contentItem</b> An ecm.model.ContentItem that represents the document that is to be opened.</p> <p><b>action</b> A string that identifies the action that the Viewer widget is to execute as “Open” or “Preview.” If the Viewer widget executes the Open action, the document is opened for editing. If the Viewer widget executes the Preview action, the document is converted to HTML and is not editable.</p>

**Receive work item event**

Description	Save the work item ID in the cache and close the currently viewed attachments to prepare for opening the attachments for another work item.
Event ID	icm.SendWorkItem



---

Payload	<p><b>workitemEditable</b> An <code>icm.model.WorkItemEditable</code> object that represents the work item for which the ID is to be saved.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>
---------	--

---

### Select case event

---

Description	Save the case ID in the cache and close the currently viewed document to prepare for opening another case.
Event ID	<code>icm.SelectCase</code>
Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case for which the ID is to be saved.</p>

---





### Send case information event

---

Description	Save the case ID in the cache and close currently viewed documents to prepare for opening documents from another case.
Event ID	<code>icm.SendCaseInfo</code>
Payload	<p><b>caseEditable</b> An <code>icm.model.CaseEditable</code> object that represents the case for which the ID is to be saved.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally by the widgets in the same page.</p>

---

#### Related concepts:

-  [Class `ecm.model.ContentItem`](#)
-  [Class `icm.model.CaseEditable`](#)
-  [Class `icm.model.WorkItemEditable`](#)
-  [Class `icm.base.Coordination`](#)

### Website Viewer widget events

The Website Viewer widget handles an incoming event to open a specified website.  
 “Website Viewer widget incoming events”

#### Website Viewer widget incoming events:

The Website Viewer widget provides an incoming event to handle the data that is received from other widgets.

#### Display URL event

---

Description	Display the website for the specified URL.
Event ID	<code>icm.DisplayUrl</code>

---

---

Payload	<b>url</b> A string that contains the URL of the website that is to be displayed.
---------	--

---

## Work Item Toolbar widget events

The Work Item Toolbar widget handles events to enable users to view and process their work items.

For example, the Work Item Toolbar widget handles an incoming event when a user adds a task to a case.

The Work Item Toolbar widget also publishes events for certain actions that you add to the widget.

“Work Item Toolbar widget incoming events”

### Work Item Toolbar widget incoming events:

The Work Item Toolbar widget uses events to communicate with other widgets.

#### Add task event

---

Description	The user selected a task type of a case to add a task that is contained in the event payload.
Event ID	<code>icm.SendNewTaskInfo</code>
Payload	<p><b>taskEditable</b> An <code>icm.model.TaskEditable</code> object that represents the task that is to be added.</p> <p><b>coordination</b> An <code>icm.util.Coordination</code> object that is used internally as a coordinator by the page widgets in the same page.</p>

---

#### Receive work item event

---

Description	Display the work item that is contained in the event payload.
Event ID	<code>icm.SendWorkItem</code>

---

---

Payload

**workItemEditable**

An `icm.model.WorkItemEditable` object that represents the work item to be opened.

**coordination**

An `icm.util.Coordination` object that is used internally by the widgets in the same page.

**UIState**

A Dojo Stateful object that is used internally by the widgets. This object can contain the following properties:

**GetNext**

This property determines whether the next work item is to be opened automatically.

**workitemReadOnly**


This property determines whether the work item is to be opened in view mode.

**GetNextCfg**

This property determines the configuration of the Open Next Work Item action.

---

**Related concepts:**

 [Class `icm.model.TaskEditable`](#)

 [Class `icm.model.WorkItemEditable`](#)

 [icm.base.Coordination](#)

---

## Testing your solution

Before you deploy your solution into production, you can test it to verify that the solution components are working correctly by deploying it into a development environment.

### About this task

You can view the solution deployment status in the solution summary view on the Manage Solutions page.

### Procedure

To validate and test your solution:

1. From the Manage Solutions page, click **Commit** to make all of your edits and updates available for deployment.
2. From the Manage Solutions page, click **Deploy** in the solution summary to deploy the solution to your project area. Deployment is a process that runs in the background. You can work on other solutions while the solution is being deployed.

If multiple users are editing the same solution, a list of locked items is shown, and you are prompted to continue to deploy the solution, or to cancel deploying the solution. If you continue to deploy the solution, it will not contain any uncommitted changes.

The **Deploy** action deploys only the artifacts that changed since the last time you deployed the solution. In some situations, you must redeploy all solution artifacts. For example, you must redeploy all solution artifacts in the following instances:

- You modified or added a globalization file, one of the Resource.js files, for the solution in the development object store.
- You updated the solution workflow file (global XPDL), but changed nothing else in the solution.

To deploy all solution artifacts, click **More Actions > Redeploy All**.

3. Click **Test** to open Case Manager Client.
  - a. The first time you test your solution expand your solution in the list of available solutions, click **Manage Roles**, and add yourself as a user to one or more roles.
  - b. Click **Add Case** to add a case to the solution. Verify that the preconditions for the case are met.
  - c. Open and complete a work item from the in-basket to verify that the page works correctly.
  - d. Verify that case types, tasks, and roles are created and working. To verify that the other components in the solution work correctly, open and test other pages and cases. You can search for cases by going to the case pages and searching for a case by date.

## What to do next

When you are ready to deploy the solution into a production environment, contact your IT administrator. The IT administrator needs the name of the solution to deploy it into production by using the IBM Case Manager administration client.

If you are done testing this solution and all other solutions in the project area, you can click **Actions > Reset Test Environment** on the Manage Solutions page.

**Attention:** When you reset the project area, all deployed solutions are removed from that project area. However, your solution definitions are not removed from the Case Manager Builder.

---

## Enhancing solution designs by using Process Designer

You can enhance your solution by using Process Designer to add primary queues, add more in-baskets, and define custom roles.

### About this task

You can open Process Designer from Case Manager Builder in the following locations:

- In the Manage Solutions page, click **More actions** > **Open Process Designer** from the solution that you want to edit. By opening Process Designer from here, you can select the case type and update the task process maps or add nontask maps, roles, in-baskets, and other items. In addition, you can open different case types without closing Process Designer.
- In the Tasks page, click the **Open Process Designer** icon for the task that you want to edit. If you open Process Designer from here you can edit the workflow for that task and other tasks in the case type. However you cannot view configuration, roles, or in-baskets.

“Primary queue”

“Creating more than one in-basket in Process Designer” on page 208

“Assigning a primary queue to more than one role” on page 209

“Disassociating a role’s primary queue” on page 210

“Defining a custom role” on page 210

“Changing primary queue definitions after upgrade” on page 211

“Adding case operations to a task” on page 212

### Related information:

Cannot access Process Designer from Case Manager Builder with Google Chrome

---

## Primary queue

To support multiple in-baskets for one role that might not be from the same role-based queue, you can use primary queues. In the Case Manager Builder Step Designer a nonsystem step in a task process map is assigned to a queue or a workgroup, not a role. When you add a step on a role swimlane, an attribute for the role indicates the primary queue that the step will be assigned to. For each role there is only one primary queue.

Only role swimlanes that have a primary queue attribute are used in Step Designer. Roles without a primary queue are ignored and cannot be added to the task process map as a swimlane. If the same primary queue is mapped to more than one role, Step Designer displays only one of the roles that matches the primary queue as an option to add as a role swimlane to the task process map.

The value for the primary queue attribute, `CB_PrimaryQueue`, is the queue name that is defined by Case Manager Builder when you define a role. The format of the value is `<solution prefix>_<normalized role name>`. For example, if you define *role A* for a solution called *My Solution* that has the solution prefix *myso*, Case Manager Builder defines a queue name of *myso\_roleA* and sets the value for the `CB_PrimaryQueue` attribute for *role A* as *myso\_roleA*.

When you upgrade a solution from a previous version of IBM Case Manager, Case Manager Builder creates the primary queue value for the role by searching for the queue of the first in-basket with the CB\_inbasket attribute in the in-basket list that is associated with the role.

Two or more roles can share the same primary queue under these conditions:

- An upgraded solution contains two or more roles that have their first in-basket (with CB\_Inbasket attribute) and those in-baskets point to the same queue
- You manually set CB\_PrimaryQueue for two roles to share the same queue in Process Designer

**Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

---

## Creating more than one in-basket in Process Designer

IBM Case Manager supports roles with multiple in-baskets enabling different views in Case Manager Client. You can define additional in-baskets and associate additional in-baskets to a role in Process Designer.

### About this task

For example, *role A* might want to see work from *role B* or the supervisor role might want to see the work items from all roles. Or, *role A* might want to see his work items in different views, for example, one view might show the status and another view might show the arrival date.

If you define *role A* for a solution called *My Solution* that has the solution prefix *myso*, Case Manager Builder defines a queue name of *myso\_roleA*. If you define *role B* for the same solution, Case Manager Builder defines a queue named *myso\_roleB*. In addition, Case Manager Builder defines a default role-based queue in-basket for both roles. After defining these roles and completing the initial solution definition in Case Manager Builder, you can create another in-basket in Process Designer so that you enable another view for *role B*.

### Procedure

To create a new in-basket in Process Designer:

1. On the Manage Solutions page click **More actions > Open Process Designer** from the solution that you want to edit.
2. Go to **View > In-baskets**.
3. In the **In-baskets** tab, select the queue from the drop down list, select *myso\_roleA*, and then add or copy the in-basket.
4. In the **Queue for in-baskets** section click the **Add** icon to create a new inbasket for queue *myso\_roleA*. Optionally, you can select an existing in-basket and click the **Copy** icon. Then, edit the copied in-basket to remove, reorder, or add properties.
5. In the Create Columns and Labels tab click the **Add** icon to add properties to display for this in-basket. You can also create filters for the in-basket.
6. Associate *role B* with the new in-basket.
  - a. Go to **View > Roles**.
  - b. Select *role B* and click the **Modify** icon in the Select In-baskets and Members tab.
  - c. Select the new in-basket and click **OK**.

This results in *role B* having access to two different queues, *myso\_roleA* and *myso\_roleB*. Even though *role B* associates with the in-basket from *role A* queue, the primary queue for *role B* is still *myso\_RoleB*. Therefore, if a step is added to the *role B* swimlane in the Step Designer, that step is assigned to *myso\_roleB* queue.

**Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

---

## Assigning a primary queue to more than one role

When you define a role, Case Manager Builder defines the `CB_PrimaryQueue` attribute for the role with value of `<solution prefix>_<normalized version of role name>`. You can assign the same primary queue to more than one role by setting the same `CB_PrimaryQueue` attribute on each role in Process Designer so that those roles share the same queue.

### About this task

The `CB_PrimaryQueue` attribute for a role is automatically defined when you create a role for a solution in Case Manager Builder. Renaming the role does not affect the queue name that was created when the role was first created in Case Manager Builder.

Step Designer displays role swimlanes, but the steps defined in each role swimlane are assigned to a primary queue of a role. By defining the same primary queue to more than one role you are enabling multiple roles to use the same work queue. In other words, all steps defined in a role swimlane where the primary queue is being shared can be processed by any role that has that same `CB_PrimaryQueue` attribute value.

### Procedure

To assign the same primary queue to more than one role:

1. In Case Manager Builder create two roles, for example, *Role 1* and *Role 2*.
2. After you create a case type, open the Task page and click **Task > New Task** to create a new task for the solution.
3. Save and close the solution.
4. In the Manage Solutions page, click **More actions > Open Process Designer** icon to open the solution in Process Designer.
5. Select the case type.
6. Assign the primary queue attribute for *Role 1* to *Role 2*.
  - a. Click **Views > Roles**.
  - b. Select *Role 1* and open the Custom Attributes tab to view the value set for the `CB_PrimaryQueue` attribute.
  - c. Select *Role 2*, open the Custom Attributes tab, update the value for the `CB_PrimaryQueue` attribute to the value that is set for *Role 1*.
- Role 2 now has the same primary queue as Role 1.
7. Go to **File > Solution**, and then save and close the solution in Process Designer.
8. In Case Manager Builder, open the solution, or ensure that the solution is in edit mode.
9. In the Task page of Case Manager Builder click the **Edit steps** icon for the new task to open the Step Designer so that you can design the task process map.

Only one of the roles is an option as a role swimlane, but all steps assigned to that swimlane are available to view by both *Role 1* and *Role 2* in the Case Manager Client.

**Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

---

## Disassociating a role's primary queue

If you have a role that will not be completing work you can disassociate the role's primary queue in Process Designer so that the role is not an option as a swimlane in the Step Designer and no work can be assigned to that role.

### About this task

For example, if the supervisor role in your solution only views work assignments but does not complete work items you can disassociate the primary queue for the supervisor role.

### Procedure

To disassociate a primary queue for a role:

1. On the Manage Solutions page click **More actions > Open Process Designer** from the solution that you want to edit.
2. Click **View > Roles**.
3. Select the role and click the **Custom Attributes** tab.
4. Select the CB\_PrimaryQueue attribute and click the **Delete** icon.
5. Optional: Assign other in-baskets to view work on the Select In-baskets and Members page by clicking the **Modify** icon.

**Related information:**

Cannot access Process Designer from Case Manager Builder with Google Chrome

---

## Defining a custom role

You can create a custom role, queue, and in-basket in Process Designer. If you want to use the role in the Step Designer, you must add the CB\_PrimaryQueue attribute for the role to indicate the primary queue for the role.

### About this task

Create a custom role and work queue if you want to create your own queue and role that does not follow Case Manager Builder naming convention or if you want to reuse a queue from the reused workflows.

### Procedure

To define a custom role, queue, and inbasket:

1. In the Manage Solutions page, select your solution, and then click **More actions > Open Process Designer**.
2. In Process Designer go to **View > Configuration > Work Queue > New** and name the new queue. For example, name the queue "Processing".
3. Expose the system fields and properties or data fields that you want displayed in this queue. Only single value properties are available for selection.



4. On the In-baskets page create an in-basket and assign the properties that you want displayed for this view. Only exposed fields for the queue are available to be used as in-basket columns and filters. Name the in-basket, for example, “My Inbox”.
5. Create a role by going to **View > Roles** and clicking the **Add** icon. For example, name the role “Adjuster”.
6. Associate the role with the in-basket.
  - a. Select the role and click the **Modify** icon on the Select In-baskets and Members page.
  - b. Select the new in-basket and click **OK**.
7. Optional: If you want to create a swimlane for this role in the Step Designer, add the CB\_PrimaryQueue attribute to the role:
  - a. Select the role, and then go to the Custom Attributes page.
  - b. Enter CB\_PrimaryQueue for the attribute name, select **string** as the attribute type, and enter Processing for the value.
8. Click **File > Solution > Save and Close** to save and close the solution.

## Results

If the solution is edited in Case Manager Builder, the role is displayed on the Roles page and the inbasket is displayed on the In-baskets page. The role is also available as a role lane choice in the Step Designer.

### Related information:

Cannot access Process Designer from Case Manager Builder with Google Chrome

---

## Changing primary queue definitions after upgrade

When upgrading a solution from a previous version of IBM Case Manager, Case Manager Builder creates the primary queue value for the role by searching for the queue of the first in-basket with the CB\_Inbasket attribute in the in-basket list that is associated to that role. The upgrade process might assign the same primary queue for different roles depending on the order of the in-baskets in the in-basket list and their CB\_Inbasket attributes. If you want each role to have a unique primary queue, you must update the CB\_PrimaryQueue attribute in Process Designer.

### About this task

You can view what primary queue value was created for each role during the upgrade in the application server log, which is named `SystemOut.log`, for IBM Case Manager, or you can see the value in Process Designer.

### Procedure

To change the primary queue definition of a role:

1. On the Manage Solutions page, click the **Open Process Designer** icon to open the solution in Process Designer.
2. Click **Views > Roles**.
3. Select the role name and open the Custom Attributes page to view the value set for the CB\_PrimaryQueue attribute.
4. Enter an existing queue name.
5. Save and close the solution in Process Designer.

### Related information:

Cannot access Process Designer from Case Manager Builder with Google Chrome

---

## Adding case operations to a task

You can use case operations to automate specific steps in a task workflow. For example, you can use case operations to create an insurance claim case when an accident report is received or to start a discretionary task based on the value of a document property.

### Before you begin

Define the task in Case Manager Builder.

### About this task

To add case operations, you open the task in Process Designer. You then add a component step to the workflow with which you associate the case operations. You can associate more than one operation to a component step. For example, you can configure case operations to create a case and add a comment with a single component step.

By adding case operations, you can perform many case-related actions, such as the following:

- Create a case using a specified case type
- Add a comment to the current case
- Create a discretionary task in a case
- File attachments to a case into a subfolder
- Relate the current case to another case

### Procedure

To add case operations to a workflow step:

1. In Case Manager Builder, click the **Open Process Designer** icon for the task to which you want to add the case operation.
2. In Process Designer, drag and drop the **Component** icon to the place in the task flow that you want the operation to be performed.
3. On the **Component** tab, click the **Add** icon.
4. From the **Component** list, select **ICM\_Operations**.
5. Select the operation that you want to insert and then click **OK**.
6. In the **Operation Parameters** area, enter an expression for each operation parameter:
  - a. Click in the **Expression** field for the parameter.
  - b. From the **Expression** list, select **Build Expression**.
  - c. For the **caseID** parameter, in the **Business Object Fields** list, you must select **F\_CaseFolder** and click **Insert**.
  - d. For the **taskID** parameter, in the **Business Object Fields** list, you must select **F\_CaseTask** and click **Insert**.
  - e. Use other case properties for the expression. In the **Business Object Fields** list, select **F\_CaseFolder**, then select the case property and click **Insert**.

If the expression contains a hard-coded string, enclose the string in double quotation marks.

“addCaseOwners operation” on page 214  
“addCaseRoleMembers operation” on page 214  
“addCommentToCurrentCase operation” on page 214  
“addCommentToCurrentDocument operation” on page 215  
“addCommentToCurrentTask operation” on page 215  
“addDependentObject operation” on page 216  
“completeCurrentCaseStage operation” on page 216  
“createCasePackage operation” on page 216  
“createCaseUsingSameCaseType operation” on page 218  
“createCaseUsingSpecifiedCaseType operation” on page 219  
“createDefaultCasePackage operation” on page 220  
“createDiscretionaryTaskInCurrentCase operation” on page 220  
“createDiscretionaryTaskInCurrentCase WithWorkflowParams operation” on page 220  
“createDiscretionaryTaskWithProps operation” on page 221  
“createNewDependentObject operation” on page 223  
“createSubfolderUnderCurrentCase operation” on page 223  
“fileAttachmentsToCurrentCase operation” on page 223  
“findDependentObjects operation” on page 224  
“getCasePropertyNames operation” on page 225  
“getCasePropertyValues operation” on page 225  
“getCaseStructure operation” on page 226  
“getTaskPropertyNames operation” on page 227  
“getTaskPropertyValues operation” on page 227  
“placeCurrentCaseStageOnHold operation” on page 227  
“relateCurrentCase operation” on page 228  
“releaseCurrentOnHoldCaseStage operation” on page 228  
“removeCaseOwners operation” on page 229  
“removeCaseRoleMembers operation” on page 229  
“removeDependentObject operation” on page 229  
“replaceCaseOwners operation” on page 229  
“replaceCaseRoleMembers operation” on page 230  
“restartPreviousCaseStage operation” on page 230  
“searchTasks operation” on page 231  
“setCasePropertyValues operation” on page 231  
“setDependentObjectProperties operation” on page 231  
“setTaskPropertyValues operation” on page 232  
“terminateTasksInCurrentCase operation” on page 232  
“unfileAttachmentFromCurrentCase operation” on page 233  
“unfileDocAttachmentFromCurrentCase operation” on page 233  
“unrelateCurrentCase operation” on page 234

## addCaseOwners operation

Assigns users to a case team as owners of the current case.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
assigner	VWParticipant	A VWParticipant object that represent the name of the user who is assigning owners to this case.
<b>owners</b>	VWParticipant[]	An array of VWParticipant objects that represent the names of the users who are to be assigned as owners of this case.

## addCaseRoleMembers operation

Assigns users to the case team as members of a role in the current case.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>roleName</b>	String	The role to which case members are to be assigned for this case.
assigner	VWParticipant	A VWParticipant object that represent the name of the user who is assigning members to the specified role in this case.
<b>members</b>	VWParticipant[]	An array of VWParticipant objects that represent the names of the users who are to be assigned to the specified role in this case.

## addCommentToCurrentCase operation

Adds a comment to the current case. You can specify the comment or use Expression Builder to assign a string case property as the comment.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>comment</b>	String	The comment that is to be added to the case.  The comment must be 64 KB or less in size.

Parameter	Type	Description
<b>return_value</b>	String	GUID of the new comment.  To set the return value, click in the <b>Expression</b> field and select <Create return value>.

## addCommentToCurrentDocument operation

Adds a new comment to the specified case document or attachment. To add a comment, the document or attachment must be filed in the current case.

The document title that is used on the comment is the short name of the released version of the document.

If a case type is configured so that users can attach documents from repositories other than the case management target object store, you can use this operation to add a comment to a document from an external repository. To add a comment to an attachment from an external repository, an external document reference object must be filed in the case along with the information that is used to reference the external document.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>doc</b>	Attachment	The document where you want to add a comment.
<b>comment</b>	String	The comment that you want to add.
<b>return_value</b>	String	The GUID of the new comment.

## addCommentToCurrentTask operation

Adds a comment to the current task. You can specify the comment or use Expression Builder to assign a string case property as the comment.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>taskId</b>	String	<b>F_CaseTask</b> must be used as parameter in Process Designer.

Parameter	Type	Description
<b>comment</b>	String	The comment to be added  The comment must be 64 KB or less in size.
<b>return_value</b>	String	The GUID of the new comment.

## addDependentObject operation

Adds a new object at the specified location in a list of dependent objects for a case property of type Business Object.

Parameter	Type	Description
<b>boPropValue</b>	String	The value of the case property to which the dependent object is to be added.  Use the <code>getCasePropertyValues</code> operation to obtain the value of the business object property.
<b>index</b>	Integer	The index at which the dependent object is to be added.
<b>propertyNames</b>	String[]	The names of the properties in the dependent object that is being added.
<b>propertyValues</b>	String[]	The values that are to be assigned to the properties in the dependent object. You must specify a value for every property that is listed in the <b>propertyNames</b> parameter.  For a multiple value property, the property value format is {'value1', 'value2' ...}.

## completeCurrentCaseStage operation

Completes the current case stage and advances to the next case stage if there is a next case stage.

When this operation completes, it returns a string that contains the name of the next case stage, which is the new current case stage. If there is no next case stage, the string contains *CmAcmComplete*.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.

## createCasePackage operation

Creates a package that contains the case artifacts that are specified by the parameters for this operation and stores the package in the case folder.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>casePackageName</b>	String	The name of the case package.
<b>locale</b>	String	The two-letter language code. The createCasePackage operation also supports the following four-letter language codes:  <b>en_US</b> English (United States)  <b>nb_NO</b> Norwegian (Bokmål)  <b>pt_BR</b> Portuguese (Brazilian)  <b>zh_CN</b> Chinese (Simplified)  <b>zh_TW</b> Chinese (Traditional)

Parameter	Type	Description
<b>packagingOptionNames</b>	String[]	<p>The list of artifacts that are to be included in the case package.</p> <p>The list can include the following values:</p> <p><b>includeAllCaseProperties</b> Set to True to include all case properties in the package.</p> <p><b>includeAllSystemProperties</b> Set to True to include all system properties in the package.</p> <p><b>includeAllCaseDocuments</b> Set to True to include all case documents in the package.</p> <p><b>includeAllCaseRelationships</b> Set to True to include all case relationships in the package.</p> <p><b>includeAllComments</b> Set to True to include all case, document, work item, and task comments in the package.</p> <p><b>includeCaseComments</b> Set to True to include all case comments in the package.</p> <p><b>includeDocumentComments</b> Set to True to include all document comments in the package.</p> <p><b>includeWorkItemComments</b> Set to True to include all work item comments in the package.</p> <p><b>includeTaskComments</b> Set to True to include all task comments in the package.</p> <p><b>includeCaseHistory</b> Set to True to include the case history in the package.</p> <p><b>includeCaseEvents</b> Set to True to include the case events in the package.</p> <p><b>Author</b> A string that specifies the name of the case package author.</p> <p><b>Title</b> A string that specifies a title for the case package.</p> <p><b>Keywords</b> A set of keywords for the case package. Use commas to separate the keywords.</p> <p><b>Subject</b> A string that specifies the subject for the case package.</p> <p>Enclose the option names in quotation marks and separate the names with commas.</p>
<b>packagingOptionValues</b>	String[]	The list of values corresponding to the options that are specified for the <b>packagingOptionNames</b> parameter.

## createCaseUsingSameCaseType operation

Creates a new instance of case using the same case type.

Before you use the createCaseUsingSameCaseType operation, use the getCasePropertyNames, getCasePropertyValues, and getCaseStructure operations to get the appropriate return values. Use these returned values for **customPropertyNames**, **customPropertyValues**, and **caseStructure** parameters in the createCaseUsingSameCaseType operation.



Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>customPropertyNames</b>	String[ ]	An array of strings that contains the names of the case properties that are to be set for the new case. You can use the values returned from the <code>getCasePropertyNames</code> operation for this parameter.
<b>customPropertyValues</b>	String[ ]	An array of strings that contains the values they are returned from the <code>getCasePropertyValues</code> operation.
<b>caseStructure</b>	String[ ]	The case structure value returned from the <code>getCaseStructure</code> operation, passed as an array of strings in the form <i>folder_path/=doc_version_series_ID</i> .
<b>return_value</b>	String	The GUID of the new case.

**Related reference:**

“`getCasePropertyNames` operation” on page 225

“`getCasePropertyValues` operation” on page 225

“`getCaseStructure` operation” on page 226

## createCaseUsingSpecifiedCaseType operation

Creates a new instance of a case for the specified case type.

Before you use the `createCaseUsingSpecifiedCaseType` operation, use the `getCasePropertyNames`, `getCasePropertyValues`, and `getCaseStructure` operations to get the appropriate return values. Use these returned values for **customPropertyNames**, **customPropertyValues**, and **caseStructure** parameters in the `createCaseUsingSpecifiedCaseType` operation.

**Tip:** The `getCasePropertyNames` and `getCasePropertyValues` operations require that you specify the identifier for an existing case. If this case is not of the same case type as you specify for the `createCaseUsingSpecifiedCaseType` operation, the property names and values might not match. The `createCaseUsingSpecifiedCaseType` operation can match values for any case properties that are shared between the two case types. Values for case properties that are not shared are blank in the new case.

Parameter	Type	Description
<b>caseType</b>	String	The case type name in symbolic form.
<b>customPropertyNames</b>	String[ ]	Values that are returned from the <code>getCasePropertyNames</code> operation.
<b>customPropertyValues</b>	String[ ]	Values that are returned from the <code>getCasePropertyValues</code> operation.
<b>caseStructure</b>	String[ ]	The case structure value that is returned from the <code>getCaseStructure</code> operation, passed as an array of strings of the form <i>folder_path/=doc_version_series_ID</i> .

Parameter	Type	Description
return_value	String	The GUID of the new case.

**Related reference:**

“getCasePropertyNames operation” on page 225

“getCasePropertyValues operation” on page 225

“getCaseStructure operation” on page 226

## createDefaultCasePackage operation

Creates a package that contains all case-related artifacts and stores the package in the case folder.

Parameter	Type	Description
caseId	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
casePackageName	String	The name of the case package.

## createDiscretionaryTaskInCurrentCase operation

Creates a new task instance of a discretionary task type. The discretionary task must be defined in the same solution.

Parameter	Type	Description
caseId	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
discretionaryTaskName	String	The symbolic name of the discretionary task. The name format must be PREFIX_TaskName.
return_value	String	The GUID of the new task.

## createDiscretionaryTaskInCurrentCase WithWorkflowParams operation

Creates a new task instance of a discretionary task type with the specified workflow parameters.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>discretionaryTaskName</b>	String	The symbolic name of the discretionary task. The name format must be PREFIX_TaskName.  This parameter is also used as the default task name. If F_Subject is specified, that is used as the task name instead (truncated to 64 characters).
<b>workflowParameter AuthoredNames</b>	String[]	An array of parameter authored names to add to the launched workflow for the discretionary task. If no properties must be set, specify a null string {""} as an empty set. F_Subject and F_Comment can be included.
<b>workflowParameterValues</b>	String[]	A String array of parameter values, if any, to add to the launched workflow for the discretionary task. Time values are expressed in the form MM/dd/yyyy HH:mm:ss. Multivalue properties are expressed in a string representation of arrays. The parameters in the inner array should be surrounded by single quotes ('), not double quotes ("). The following example shows a possible entry in Process Designer for this argument:  <pre>{ "123", "auto",   "{ 'one', 'two',     'three' }",   "medical" }</pre> Process Engine-specific objects, such as Attachments or participants, can be passed into this argument, as they will be converted into a string automatically. If no properties need to be set, specify a null string {""} as an empty set. To specify an empty attachment, specify "    0   0    " for a single attachment and "{     0   0     }" for an attachment array.
<b>return_value</b>	String	The GUID of the new task.

## createDiscretionaryTaskWithProps operation

Creates a new task instance of a discretionary task type with the specified properties.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.

Parameter	Type	Description
<b>discretionaryTask SymbolicName</b>	String	The symbolic name of the discretionary task. The name format must be PREFIX_TaskName.  This parameter is also used as the default task name. If F_Subject is specified, that is used as the task name instead (truncated to 64 characters).
<b>taskPropertyNames</b>	String[]	A String array of task property symbolic names. If no properties need to be set, specify an empty string array {""}.
<b>taskPropertyValues</b>	String[]	A String array of property values. The list length must have the same length as the <b>taskPropertyNames</b> parameter. Time values are expressed in the form yyyy-MM-ddTHH:mm:ssZ. Multivalue properties are expressed in a string representation of arrays. The parameters in the inner array should be surrounded by single quotes ('), not double quotes ("). The following example shows a possible entry in Process Designer for this argument: <pre>{ "123", "auto",   "{ 'one', 'two',     'three' }",   "medical" }</pre>
<b>workflowParameter AuthoredNames</b>	String[]	An array of parameter authored names to add to the launched workflow for the discretionary task. If no properties must be set, specify a null string {""} as an empty set. F_Subject and F_Comment can be included.
<b>workflowParameter Values</b>	String[]	A String array of parameter values, if any, to add to the launched workflow for the discretionary task. Time values are expressed in the form MM/dd/yyyy HH:mm:ss. Multivalue properties are expressed in a string representation of arrays. The parameters in the inner array should be surrounded by single quotes ('), not double quotes ("). The following example shows a possible entry in Process Designer for this argument: <pre>{ "123", "auto",   "{ 'one', 'two',     'three' }",   "medical" }</pre> Process Engine-specific objects, such as Attachments or participants, can be passed into this argument, as they will be converted into a string automatically. If no properties need to be set, specify a null string {""} as an empty set. To specify an empty attachment, specify " 0 0 0 0 " for a single attachment and "{ " 0 0 0 0 " }" for an attachment array.
<b>return_value</b>	String	The GUID of the new task.

## createNewDependentObject operation

Creates a pending instance of the specified dependent object class to use as the starting value for a new case property. If you want, you can then use `addDependentObject` operation to add more values. You assign the collected values as the value of a business object property when you create a case.

Parameter	Type	Description
<b>requiredClassName</b>	String	The name of the dependent object class from which this instance is to be created.
<b>propertyNames</b>	String[]	The names of the properties in the new dependent object instance.
<b>propertyValues</b>	String[]	The new values that are to be assigned to the properties in the new dependent object instance. You must specify a value for every property that is listed in the <b>propertyNames</b> parameter.  For a multiple value property, the property value format is {'value1', 'value2' ...}.

## createSubfolderUnderCurrentCase operation

Creates a subfolder under the current case folder.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>subfolderPath</b>	String	The name of the subfolder to be created under the current case folder.
<b>return_value</b>	VWAttachment	The newly-created subfolder, which is returned as an attachment object.

## fileAttachmentsToCurrentCase operation

Files attachments to a specified subfolder under the current case folder. Attachments can originate in the case management object stores or in other repositories that are configured for the case management environment.

You can use this operation only to file attachments to an existing subfolder. To file an attachment in a new subfolder, you must first call the `createSubfolderUnderCurrentCase` operation to create the subfolder.

If a case type is configured so that users can attach documents from repositories other than the case management target object store, you can use this operation to file attachments from an external repository. For an attachment from an external repository, this operation files the external document reference object in the case along with the information that is required to reference the external document.

If a case type does not support external repositories, you can use the FileAttachmentsToCurrentCase operation only to file attachments that are in the same target object store as the current case.

If an attachment cannot be found, the system stops execution of the current work item and returns an error.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>attachments</b>	VWAttachment[]	An array of VWAttachment objects, which are the attachments for the filing document. Attachments can originate from the case management repository or other repositories.
<b>subfolderPath</b>	String	Path to the subfolder, relative to the Case folder. If the documents to be filed are in the root of the case folder, this path can be the empty string or "/".
<b>return_value</b>	VWAttachment	The subfolder, which is returned as a VWAttachment object.

**Related reference:**

“createSubfolderUnderCurrentCase operation” on page 223

## findDependentObjects operation

Finds the indexes of objects that match the specified criteria in a list of dependent objects for a case property of type Business Object.

Parameter	Type	Description
<b>boPropValue</b>	String	The value of the case property in which the dependent objects are to be found.  Use the <code>getCasePropertyValues</code> operation to obtain the value of the business object property.
<b>propertyNames</b>	String[]	The names of the properties that are to be searched for.

Parameter	Type	Description
<b>propertyValues</b>	String[]	<p>The values of the properties that are to be searched for. You must specify a value for every property that is listed in the <b>propertyNames</b> parameter.</p> <p>If you are searching for more than one property, the findDependentObjects operation applies the AND operator to the name-value pairs.</p> <p>This operation supports only exact matches for both single value and multivalued properties.</p> <p>For the multiple value property, enter the property value in the following format: {'value1', 'value2'...}. A match occurs only if the property has the same number of values in the same order.</p>

## getCasePropertyNames operation

Returns a list of the names of the properties in a case. This operation is a prerequisite for the createCase operations.

Parameter	Type	Description
<b>caseId</b>	String	<p>The identifier of the case.</p> <p>You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.</p>
<b>caseType</b>	String	The case type to be used.
<b>return_value</b>	String[]	The CasePropertyNames value is an array of strings, containing the property names of the case. This return value is used by the createCase operations and the getCasePropertyValues operation.

### Related reference:

“createCaseUsingSameCaseType operation” on page 218

“createCaseUsingSpecifiedCaseType operation” on page 219

## getCasePropertyValues operation

Returns a list of the property values of a specified case. This operation is a prerequisite for the createCase operations.

**Tip:** You can use the values that are returned by this operation to set values for a new case. If the new case is not of the same case type as you specify for the getCasePropertyValues operation, the property names and values might not match.

IBM Case Manager can match values only for case properties that are shared between the two case types. Values for case properties that are not shared are blank in the new case.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>customPropertyNames</b>	String[]	Use the return value from the <code>getCasePropertyNames</code> operation.
<b>return_value</b>	String[]	The <code>CasePropertyValues</code> value is an array of strings, containing the property values of the requested properties. This return value is used by the <code>createCase</code> operations.

**Related reference:**

“`createCaseUsingSameCaseType` operation” on page 218

“`createCaseUsingSpecifiedCaseType` operation” on page 219

## getCaseStructure operation

Returns the case structure for a given case. The structure is returned as a list of folder names and document series IDs, but excludes any empty folders. Typically, this operation is called to obtain the case structure prior to calling the `createCase` operations.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>return_value</b>	String[]	The <code>CaseStructure</code> value is an array of strings, containing the case structure as strings of the form <i>folder_path/</i> <i>=doc_version_series_ID.</i>

**Related reference:**

“`createCaseUsingSameCaseType` operation” on page 218

“`createCaseUsingSpecifiedCaseType` operation” on page 219



## getTaskPropertyNames operation

Returns the custom property names of a task.

Parameter	Type	Description
<b>taskId</b>	String	This task ID value can be the ID of the current task, which is obtained by passing F_CaseTask, or an explicit ID of another task.
<b>return_value</b>	String[]	The task property names in a string array.

## getTaskPropertyValues operation

Returns task property values for a given list of task properties.

Parameter	Type	Description
<b>taskId</b>	String	The task ID, which can be the current task that is obtained by passing the F_CaseTask, or an explicit ID of another task.
<b>taskPropertyNames</b>	String[]	The list of task properties to retrieve.
<b>return_value</b>	String[]	The TaskPropertyValues value is an array of strings, containing the property values of the requested properties.

## placeCurrentCaseStageOnHold operation

Puts the current case stage on hold.

When this operation completes, it returns a string that contains the name of the current case stage.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.

## relateCurrentCase operation

Creates a relationship between the current working case and the targeted case.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>relateTargetId</b>	String	For this value, use the returned value of the createCase operation that you used.
<b>relateDescription</b>	String	The description of this relationship.
<b>relationshipCategory</b>	String	The relationship category.  Optional. For related relationship types only.  Relationship categories are user-defined. A category provides additional information to identify the nature of the relationship and to facilitate searches for certain types of relationships.
<b>twoWayRelationship</b>	Boolean	A Boolean field that indicates whether the relationship is two-way.
<b>return_value</b>	String	The identifier of the new relationship.

## releaseCurrentOnHoldCaseStage operation

Releases the hold on the current case stage.

When this operation completes, it returns a string that contains the name of the current case stage.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.

## removeCaseOwners operation

Removes users as owners of the current case.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>owners</b>	VWParticipant[]	An array of VWParticipant objects that represent the names of the users who are to be removed as owners of this case.

## removeCaseRoleMembers operation

Removes users as members of a role in the current case.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>roleName</b>	String	The role from which case members are to be removed for this case.
<b>members</b>	VWParticipant[]	An array of VWParticipant objects that represent the names of the users who are to be removed from the specified role in this case.

## removeDependentObject operation

Removes the object at the specified index from a list of dependent objects for a case property of type Business Object.

Parameter	Type	Description
<b>boPropValue</b>	String	The value of the case property from which the dependent object is to be removed.  Use the <code>getCasePropertyValues</code> operation to obtain the value of the business object property.
<b>index</b>	Integer	The index of the dependent object that is to be removed.  Use the <code>findDependentObjects</code> operation to get the index of the dependent object.

## replaceCaseOwners operation

Replaces the users currently assigned as owners of the current case with a new set of users.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
assigner	VWParticipant	A VWParticipant object that represent the name of the user who is replacing owners in in this case.
<b>owners</b>	VWParticipant[]	An array of VWParticipant objects that represent the names of the users who are to replace the existing owners of this case.

## replaceCaseRoleMembers operation

Replaces the users currently assigned as members of a role in the current case with a new set of users.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>roleName</b>	String	The role in which case members are to be replaced for this case.
assigner	VWParticipant	A VWParticipant object that represent the name of the user who is replacing members of the specified role in this case.
<b>members</b>	VWParticipant[]	An array of VWParticipant objects that represent the names of the users who are to replace the existing members of the specified role in this case.

## restartPreviousCaseStage operation

Restarts the previous case stage.

When this operation completes, it returns a string that contains the name of the previous case stage, which is the new current case stage. If there is no previous case stage, the string contains the name of the current case stage.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.

## searchTasks operation

Searches for tasks in a particular case by using the provided conditions.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>taskTypes</b>	String[ ]	The list of task types to search. If an empty array is provided, all task types are searched.
<b>propertyFilter</b>	String	If not empty, this parameter specifies the property filter condition that is to be applied to the search. The string must conform to the Content Platform Engine SQL syntax, for example, ([prop1] = 1 AND [prop2] <> 3). The properties that are included in this filter must also be included in the properties parameter.
<b>return_value</b>	String[ ]	The task IDs in a string array.

## setCasePropertyValues operation

Sets the values for the specified properties in a case.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>casePropertyNames</b>	String[]	The list of case properties to update.
<b>casePropertyValues</b>	String[]	The list of property values. The list length must have the same length as casePropertyNames.

## setDependentObjectProperties operation

Updates the property values for the object at the specified index in the list of dependent objects for a case property of type Business Object.

Parameter	Type	Description
<b>boPropValue</b>	String	The value of the case property in which the dependent object is to be updated.  Use the <code>getCasePropertyValues</code> operation to obtain the value of the business object property.
<b>index</b>	Integer	The index of the dependent object that is to be updated.  Use the <code>findDependentObjects</code> operation to get the index of the dependent object.

Parameter	Type	Description
<b>propertyNames</b>	String[]	The names of the properties in the dependent object that are to be updated. You can specify a subset of the dependent object properties.
<b>propertyValues</b>	String[]	The new values to be assigned to the specified properties in the dependent object. You must specify a value for every property that is listed in the <b>propertyNames</b> parameter.  For a multiple value property, the property value format is {'value1', 'value2'...}.

## setTaskPropertyValues operation

Sets the task property values for a particular task. This method typically applies when updating the task properties on a task other than the current one. For the current task, you can use step parameters to set the task properties.

Parameter	Type	Description
<b>taskId</b>	String	The explicit ID of a task other than the current one. For the current task, use step parameters.
<b>taskPropertyNames</b>	String[]	The list of task properties to update.
<b>taskPropertyValues</b>	String[]	The list of property values. The list length must have the same length as the <b>taskPropertyNames</b> parameter. Time values are expressed in the form yyyy-MM-ddTHH:mm:ssZ. Multivalue properties are expressed in a string representation of arrays. The parameters in the inner array should be surrounded by single quotes ('), not double quotes ("). The following example shows a possible entry in Process Designer for this argument:  <pre>{ "123", "auto",   "{ 'one', 'two',   'three' }",   "medical" }</pre>

## terminateTasksInCurrentCase operation

Terminates all workflows (tasks) in the current case. Tasks are terminated only if they have not completed and have not failed.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.

## unfileAttachmentFromCurrentCase operation

Unfiles an attachment from a case folder.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>subfolderPath</b>	String	The path to the subfolder.
<b>documentTitle</b>	String	The name of the document to be unfiled. You can use the <b>CE_Operations.getStringProperty</b> operation on the document to use for this value.

## unfileDocAttachmentFromCurrentCase operation

Unfiles the specified document attachment from the folder or subfolder of the specified case. Upon successful completion of this method, the document is no longer filed in the folder.

If a case type is configured so that users can attach documents from repositories other than the case management target object store, you can use this operation to unfile attachments from an external repository. To unfile an attachment from an external repository, the attachment must be filed in the current case along with the information that is needed to reference the external document.

**Tip:** Multiple requests to unfile the document in the same folder are ignored, although the requests appear to be successful. If the request is ignored, the attachment that is returned is empty. Otherwise, the attachment that is returned is the folder or subfolder from which the document was unfiled.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.

Parameter	Type	Description
<b>subfolderPath</b>	String	Path to the subfolder, relative to the Case folder. If the document to be unfiled is in the root of the case folder, this can be the empty string or "/".
<b>sourceDocument</b>	Attachment	An Attachment object for the document attachment to unfile.
<b>return_value</b>	Attachment	The folder or subfolder from which the attachment was unfiled, or an empty attachment if the attachment was not unfiled.

## unrelateCurrentCase operation

Removes the specified relationship to the current case.

Parameter	Type	Description
<b>caseId</b>	String	The identifier of the case.  You must set the expression for the <b>caseID</b> parameter to the <b>F_CaseFolder</b> business object field in the Expression Builder window.
<b>relationshipID</b>	String	The ID of the relationship to remove.
<b>unrelateDescription</b>	String	A description for the unrelating operation on the case.



---

## 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

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation  
IBM Director of Licensing  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
USA*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample

programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. 2010, 2017. All rights reserved.

---

## Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

---

## Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

### Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

### Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

### Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these

publications or any portion thereof outside your enterprise, without the express consent of IBM.

## **Rights**

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

---

## **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](http://www.ibm.com/privacy) and IBM's Online Privacy Statement at [www.ibm.com/privacy/details](http://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](http://www.ibm.com/software/info/product-privacy).

---

# Index

## A

- AND-split
  - valid map 61
- attachments 50
- Attachments widget
  - adding documents by using entry templates 96
  - configuring 94
  - incoming events 141
  - outgoing events 140
  - overview 93

## B

- business process 68
- business process management
  - integrating 67
  - reusing processes 67
- Business Process Manager Advanced tasks 71
- business processes 38
  - adding 70
- business rules
  - adding 33
  - mapping custom parameters 59
  - setting completion menu options 37
  - table-based 33
  - text-based 33

## C

- case folder 57
- case folders 19
- Case Information widget
  - adding documents by using entry templates 96
  - incoming events 145
  - outgoing events 143
  - overview 97
  - views
    - documents 100
    - history 101
    - summary 101
    - Tasks 99
- Case List widget
  - incoming events 148
  - outgoing events 147
  - overview 101
- case management application
  - modify 76
- case management applications
  - components
    - widgets 91
- Case Manager Builder
  - multiple users 74
- case properties
  - deleting 6
- Case Stages widget
  - overview 103
- case summary
  - customizing view 21
- case title
  - setting property for 21
- Case Toolbar widget
  - incoming events 152
  - overview 103
- case types
  - adding 17
  - adding business rules 33
  - defining properties views 23
  - modifying 17
  - overview 19
  - views 21
- case workers
  - roles 7, 8
  - solution pages 8
- Chart widget
  - incoming events 153
  - overview 105
- commit solution 74
- connectors
  - drawing 60
- containers
  - height 25
  - Layout 24
  - layouts 25
  - Multiple Column Layout 24
  - Property List 24
  - Property Table 24
  - Tabbed Layout 24
  - Titled Layout 24
  - width 25
- Content List widget
  - configuring 107
  - outgoing vents 154
  - events 154
  - overview 105
  - preparing to use 106
- custom pages
  - adding to steps 60
- custom rule parameters
  - defining 33
  - mapping to external data sources 59
- custom tasks
  - overview 40

## D

- data fields 57
- decision tables 33
- discretionary tasks
  - overview 40
- document classes 13
  - adding 11
  - modifying 11
- document precondition 50
- document properties 13
- documents view
  - Case Information widget 100

## E

- Editor settings
  - Properties View Designer 28
- events
  - Content List widget 154
  - Form widget 157
  - Page Container widget 165
  - Search widget 187
  - Timeline Visualizer widget 197
  - Toolbar widget 202
  - wiring 128
- external properties
  - adding to views 28

## F

- FileNet P8 Process Designer 45
- Form widget
  - configuring form attachments 111
  - configuring form templates 110
  - events 157
  - outgoing events 156
  - overview 108

## H

- History view
  - Case Information widget 101

## I

- IBM Business Process Manager
  - adding tasks 70
- in-baskets
  - adding 11
  - multiple 209
  - primary queues 207, 209
  - Process Designer 208
- In-baskets widget
  - configuring 113
  - incoming events 161
  - outgoing events 159
  - overview 112
- incoming events
  - Attachments widget 141
  - Case Information widget 145
  - Case List widget 148
  - Case Toolbar widget 152
  - Chart widget 153
  - In-baskets widget 161
  - Instruction widget 164
  - Markup widget 164
  - Original Case Properties widget 164
  - Page Container widget 169
  - Process History widget 172
  - Properties widget 182
  - Script Adapter widget 185
  - Search widget 186
  - Split Case Properties widget 195

- incoming events (*continued*)
  - Timeline Visualizer widget 198
  - To-Do List widget 200
  - Viewer widget 202
  - Website Viewer widget 203
  - Work Item Toolbar widget 204
- initiating attachment 50
- Instruction widget
  - incoming events 164
  - overview 114

## J

- joins
  - routing 64

## M

- maps
  - rules for valid maps 61
- Markup widget
  - incoming events 164
- modify
  - case management application 76
- Multiple Column
  - container 24

## O

- Original Case Properties widget
  - incoming events 164
  - overview 116
- outgoing events
  - Attachments widget 140
  - Case Information widget 143
  - Case List widget 147
  - Content List widget 154
  - Form widget 156
  - In-baskets widget 159
  - Page Container widget 165
  - Properties widget 173
  - Script Adapter widget 185
  - Search widget 185
  - Split Case Properties widget 188
  - Timeline Visualizer widget 197

## P

- Page Container widget
  - events 165
  - incoming events 169
  - outgoing events 165
- pages 78
  - creating 76
- preconditions 50
  - tasks 43
- primary queue
  - disassociating 210
- primary queues
  - description 207
  - roles 207
  - swimlanes 207
- process 70
- Process Designer 58
  - enhancing solutions 207

- Process Designer (*continued*)
  - inbasket 208
- Process History widget
  - incoming events 172
  - overview 117
- processes 67
  - adding steps 45
- properties 6, 69
  - adding 4
  - mapping 57
  - reusing 4
- properties view
  - adding external properties 28
  - adding workflow data fields 27
  - adding workgroups 27
  - control settings 28
  - defining 23
- Properties widget
  - customizing layouts 21
  - defining layouts for 23
  - incoming events 182
  - outgoing vents 173
  - overview 117
- property editors
  - Integer and DateTime fields 29
  - settings 29
  - width 29
- Property List
  - container 24
- Property Table
  - container 24

## Q

- queue definitions
  - changing 211

## R

- required tasks
  - overview 40
- reusing processes 69, 70
- roles
  - adding 7, 8
  - assigning 7
  - assigning pages 8
  - case workers 7
  - custom 210
  - primary queues 207
  - workgroups 48, 49
- routes
  - drawing connectors 60

## S

- Script Adapter widget
  - incoming events 185
  - outgoing events 185
  - overview 120
- search
  - customizing view 21
- Search widget
  - incoming events 186
  - outgoing events 185
  - overview 122

- Select Case Documents widget
  - events 187
  - overview 123
- sets 38, 40
- solution
  - adding 1
  - committing 74
  - deploying 1
  - validating 74
- solutions
  - adding 3
  - components
    - pages 78
  - deploying 205
  - solution templates 3
  - solution wizard 3
  - testing 205
- Split Case Properties widget
  - eoutgoing vents 188
  - incoming events 195
  - overview 123
- splits
  - routing 64
- Step Designer
  - joins 64
  - routing 64
  - splits 64
- step routing
  - AND-join 61
  - AND-split 61
  - join 61
  - OR-join 61
  - OR-split 61
  - split 61
- steps 38, 50, 53
  - adding 52
  - creating 45
  - custom pages 60
  - data fields 57
- Summary view
  - Case Information widget 101
- swimlanes 51
  - adding 50
  - custom 210
  - primary queues 207

## T

- Tabbed Layout
  - container 24
- Task view
  - Case Information widget 99
- tasks 19, 45, 50, 52, 53
  - adding 38, 70, 71
  - building steps in Process Designer 58
  - custom 40
  - defining views for to-do tasks 73
  - designing 44
  - discretionary 40
  - madding 68
  - mapping custom parameters in rule steps 59
  - optional 40
  - preconditions 43
  - required 40

- tasks *(continued)*
  - running on Business Process Manager
    - Advanced 71
    - states 45
    - workflow 38, 68
    - workflow design 44
    - workflows
      - adding 70
  - Timeline Visualizer widget
    - events 197
    - incoming events 198
    - outgoing events 197
    - overview 123
  - Titled Layout
    - container 24
  - To-do List widget
    - defining layouts for 73
  - To-Do List widget
    - incoming events 200
    - overview 124
  - to-do task view
    - defining 73
  - Toolbar widget
    - events 202
    - overview 125

## U

- upgrade
  - primary queues 211

## V

- valid map 61
- validate solution 74
- Viewer widget
  - displaying documents 125
  - incoming events 202
  - overview 125
- views 19
  - case types 21

## W

- Website Viewer widget
  - incoming events 203
  - overview 126
- widgets 91
  - Attachment
    - adding documents by using entry
      - templates 96
    - configuring 94
    - incoming events 141
    - outgoing events 140
    - overview 93
  - Case Information
    - adding documents by using entry
      - templates 96
    - documents view 100
    - History view 101
    - incoming events 145
    - outgoing events 143
    - overview 97
    - Summary view 101
    - Tasks view 99

- widgets *(continued)*
  - Case List
    - incoming events 148
    - outgoing events 147
    - overview 101
  - Case Stages
    - overview 103
  - Case Toolbar
    - incoming events 152
    - overview 103
  - Chart
    - incoming events 153
    - overview 105
  - Content List
    - configuring 107
    - events 154
    - outgoing events 154
    - overview 105
    - preparing to use 106
  - Form
    - configuring form attachments 111
    - configuring form templates 110
    - events 157
    - outgoing events 156
    - overview 108
  - In-baskets
    - configuring 113
    - incoming events 161
    - outgoing events 159
    - overview 112
  - Instruction
    - incoming events 164
    - overview 114
  - Markup
    - incoming events 164
  - Original Case Properties
    - overview 116
  - Original Case Properties widget
    - incoming events 164
  - Page Container
    - events 165
    - incoming events 169
    - outgoing events 165
  - Process History
    - incoming events 172
    - overview 117
  - Properties
    - incoming events 182
    - outgoing events 173
    - overview 117
  - Script Adapter
    - incoming events 185
    - outgoing events 185
    - overview 120
  - Search
    - events 187
    - incoming events 186
    - outgoing events 185
    - overview 122
  - Select Case Documents
    - overview 123
  - Split Case Properties
    - incoming events 195
    - outgoing events 188
    - overview 123
  - Timeline Visualizer
    - events 197

- widgets *(continued)*
  - Timeline Visualizer *(continued)*
    - incoming events 198
    - outgoing events 197
    - overview 123
  - To-Do List
    - incoming events 200
    - overview 124
  - Toolbar
    - events 202
    - overview 125
  - Viewer
    - displaying documents 125
    - incoming events 202
    - overview 125
  - Website Viewer
    - incoming events 203
    - overview 126
  - wiring events 128
  - Work Item Toolbar
    - incoming events 204
    - overview 126
  - wiring events 128
  - Work Item Toolbar widget
    - incoming events 204
    - overview 126
  - workflow 51, 53, 70
    - importing 69
    - reusing 69
  - workflow data fields
    - adding to views 27
  - workflows 67
    - adding 67
    - building steps in Process
      - Designer 58
      - mapping custom parameters in rule steps 59
  - workgroups
    - adding 48, 49
    - roles 48, 49









Product Number: 5725-A15

SC19-3274-10

