



# Cúram 8.1.3

## Advisor Configuration Guide



## Note

---

Before using this information and the product it supports, read the information in [Notices on page 27](#)



# Edition

---

This edition applies to Cúram 8.1, 8.1.1, 8.1.2, and 8.1.3.

© Merative US L.P. 2012, 2024

Merative and the Merative Logo are trademarks of Merative US L.P. in the United States and other countries.



# Contents

---

|  |            |
|--|------------|
| <b>Note.....</b>   | <b>iii</b> |
| <b>Edition.....</b>  | <b>v</b>   |
| <b>1 Configuring Advisor.....</b>                            | <b>9</b>   |
| 1.1 Configuring an advice context.....                       | 9          |
| 1.2 Creating an advisor rule set.....                        | 10         |
| Advice context rule classes and rule attributes.....         | 11         |
| Advice item rule class and rule attributes.....              | 12         |
| Creating a parameter to be displayed in the advice text..... | 14         |
| Creating a link to be displayed in the advice text.....      | 14         |
| Creating the advice text.....                                | 14         |
| 1.3 Managing evidence rule objects.....                      | 15         |
| Active/in-edit succession set rule objects.....              | 15         |
| Configuration.....   | 16         |
| Conversion processing.....                                   | 17         |
| Propagation processing.....                                  | 20         |
| 1.4 Example Advisor configuration.....                       | 22         |
| Create an Advisor rule set using the CER editor.....         | 23         |
| Configure the Advice context and Advice text.....            | 25         |
| <b>Notices.....</b>  | <b>27</b>  |
| Privacy policy.....  | 28         |
| Trademarks.....  | 28         |





# 1 Configuring Advisor

---

Configure Cúram Advisor to display appropriate context-sensitive advice to users. Advisor rule sets contain rule classes rule items and rule attributes. You can configure the advice context, which defines the context in which advice is shown. The rule object converter converts evidence data into rule objects.

Advisor is a configurable infrastructure which:

- Allows CER rules to be written to analyze the state of data in the database.
- Presents the output of the analysis to users in their relevant context

To configure the Advisor, define a new advice context as part of application administration, associate relevant application pages in the application with that context. Then, associate preconfigured rule sets that determine the appropriate advice for that context, and configure the rule object converter and propagator. This results in the display of appropriate context-sensitive advice to users at run time.

## 1.1 Configuring an advice context

---

The advice context defines the context in which advice appears.

### About this task

Each context contains a set of application pages and page parameters that make up the context. Each context also contains a set of rule sets that determine the advice that appears in that context. When the advice context is configured, the system displays appropriate contextual advice that relates to the application page the user is on.

### Procedure

1. Define the advice context.
  - a) Log in to the Administration Application.
  - b) In the **Shortcuts** panel for the **Administration Workspace**, select the **User Interface** link.
  - c) Select the **Advisor** link to open the **Advisor list** page.
  - d) Select the **New Advice Context** link to open the **New Advice Context Configuration** dialog.
  - e) In the **New Advice Context Configuration** dialog, enter a unique name for the advice context. The name is the logical identifier for the advice context.
  - f) Select **Save**.
2. Add one or more pages to each advice context. Add the names of the application pages on which advice that relates to the advice context appears.
  - a) Select the **New Advice Context Key** menu item to open the **Add Advice Key** dialog.
  - b) Enter the name of the page that is being added to the advice context. This name is the *PAGE\_ID* from the related UIM file, for example, *Person\_homePage*.

- c) Select **Page** from the **Type** drop-down list. For compliance reasons, the code table for the **Advice Context Key Type** is not customizable.
  - d) Select **Save**.
3. Add parameters to pages.

Page content is determined by both the page and the parameters that drive the page. For example, to display advice on the **Person** home page, the *Person\_home* page ID is specified when that page is added to the advice context. However, to make the advice truly context sensitive, you must also add the page parameter for the **Person** home page, the page parameter is the *concernRoleID*. *concernRoleID* enables the advice that is displayed at the case level to be truly context-sensitive by narrowing the advice context to apply to the home page for a specific person.

- a) Select the **New Page Parameter** menu item to open the **New Parameter** dialog.
  - b) Enter the name of the parameter, for example *concernRoleID*.
  - c) Select the parameter type from the **Type** drop-down list, for example, number, date, or string.
  - d) Select **Save**.
4. Assign rule sets to an advice context.

Assign one or more rule sets to each advice context so that the system can display context-sensitive advice to the user. These are the rule sets that calculate the advice that relates to the pages assigned to the advice context. To assign a rule set to an advice context, you must first create the rule sets by using the CER Editor. For more information, see [1.2 Creating an advisor rule set on page 10](#).

- a) Select the **Add Rule Set** menu item.
- b) Select the appropriate rule set from the drop-down list of available rule sets.
- c) Select **Save**.

## 1.2 Creating an advisor rule set

---

To create an advisor rule set, you must define a rule set and then the advisor rules classes that are contained in the rule set must then be developed by using the CER Editor.

To create a new rule set as part of application administration, follow the steps for creating a new rule set described in Section 2.2 of the *Working With Cúram Express Rules* guide. When the rule set is created, advice rules classes that are contained in the rule set must be created by using the CER Rules Editor. The following sections describe the rule classes that must be created.

## Advice context rule classes and rule attributes

Each advice rule set must contain a single rule class, which inherits from *Abstract Advice Context*.

### Create an advice context rule class

#### About this task

This rule class is the root rule class for the rule set and is the rule class that produces the advice items that are displayed at the case level.

To create an advice context rule class:

#### Procedure

1. Create the rule class by using the CER Editor.
2. Give the rule class an appropriate name.
3. Specify that the rule class inherits from *AbstractAdviceContext* defined in the rule set that is named *CoreAdvisorRuleSet*.

#### Rule attributes

##### Mandatory rule attributes

Table 1 describes the mandatory rule attribute that must be specified for each advice context rule class:

Table 1: Mandatory Rule Attribute

This table describes the mandatory rule class attribute that must be specified.

| Rule Attribute Name | Attribute Type | Purpose   |
|---------------------|----------------|---|
| advice              | List           | Allow the rule set developer to specify the advice items that this rule set produces. The rule set developer must specify the derivation of this attribute to be a list of rule objects where the rule class is a subclass of abstract advice item. |

##### Other rule attributes

Each advice context rule class also defines a field for every page parameter that is passed in from the page. For example, if the *AdviceContext* is related to the **PersonHome** page, then the page parameter is the *concernRoleID*. To produce advice that is relevant to a specific person, the rule set must have the concern role ID available to it during processing. This is achieved by creating a rule attribute of type *NumberParameter* with the name of the page parameter on the advice context rule class. Provided the attribute has the same name as the page parameter, the Advisor infrastructure automatically detects the presence of the attribute and populates it with the correct value at run time.

## Advice item rule class and rule attributes

To create an advice item for the list, create a rule class that defines the advice item, then add an instance of the rule class to the list of advice on the advice context.

### ***Creating an advice item rule class***

#### **About this task**

The purpose of each advice context is to produce a list of advice items that are associated with the context.

#### **Procedure**

1. Create the rule class by using the CER Editor.
2. Give the rule class an appropriate name.
3. Specify that the rule class inherits from *AbstractAdviceItem* defined in the rule set named *CoreAdvisorRuleSet*.

#### **Rule attributes**

##### **Mandatory rule attributes**

The following table describes the mandatory rule attributes for each advice item rule class:

*Table 2: Mandatory Rule Attributes*

This table describes the mandatory rule attributes that must be specified.

| Rule Attribute Name | Attribute Type | Purpose   |
|---------------------|----------------|---|
| adviceText          | String         | Allows the rule set developer to link to the text that appears if this advice item is triggered. The value that this attribute contains is not the actual advice text. The value of this attribute is the name of the property entry that contains the advice text. For more information, see <a href="#">Creating the advice text on page 14</a> .                             |
| showAdvice          | Boolean        | Allows the rule set developer to specify the conditions under which this advice item is displayed. If the value is calculated as TRUE, the advice is displayed, if it is calculated as FALSE, the advice is not displayed. These rules are the main rules for the advice item, and can examine any data that is related to the context to decide whether to display the advice. |
| category            | CodeTableItem  | Allows the rule set developer to specify the category of advice that this advice belongs to. Two available categories are provided: <i>issue</i> and <i>reminder</i> .  |
| priority            | CodeTableItem  | Allows the rule set developer to specify the priority of this advice item. The priority determines the order in which the advice item appears on the list of advice that is displayed for the page.   |

| Rule Attribute Name | Attribute Type | Purpose   |
|---------------------|----------------|---|
| status              | CodeTableItem  | Allows the rule set developer to indicate whether the advice has action that is taken on it or not. The status of the advice can be <i>pending</i> or <i>complete</i> . |

### Optional rule attributes

The following table describes the optional rule attributes that can be specified for each advice item rule class:

Table 3: Optional Rule Attributes

This table describes the optional rule attributes that you can specify.

| Rule Attribute Name | Attribute Type | Purpose  |
|---------------------|----------------|--|
| caseID              | Number         | Allows the rules developer to specify whether the advice item is related to a particular case. Advisor includes a special 'awareness' of cases and evidence facility. If a caseID is specified in this attribute, the advice item is automatically linked to the case.   |
| evidenceType        | CodeTableItem  | Allows the rules developer to specify whether the advice item is related to a particular evidenceType on a particular case. It is important to note that unless a caseID is specified, this attribute is disregarded by Advisor processing. However, if this attribute is specified and if the <i>caseID</i> is also specified, and if this advice item is of category 'Issue' then this advice item appears on the Issues list for the specified evidence type when viewing the case. |
| relatedID           | Number         | Allows the rules developer to specify whether the advice item is related to a particular evidence record. Unless both a case and an evidence type are specified, this attribute is disregarded by Advisor processing. However, if this attribute is specified and if a case and evidence type is also specified, and if the advice item is of type <i>Issue</i> then this advice item appears on the issues list for the specified evidence record when viewing the case.              |
| expiryDateTime      | DateTime       | The date and time upon which this advice expires. If a piece of advice is no longer applicable from a particular point in time, then this attribute is used to specify the date from which it becomes inapplicable. If this attribute is not specified, the advice is considered to be applicable until the <i>showAdvice</i> attribute described in <a href="#">Advice item rule class and rule attributes on page 12</a> evaluates to false.   |

## Creating a parameter to be displayed in the advice text

To create an parameter that can be displayed in the advice text, you must create an attribute on the *AdviceItem* of type *StringParameter*, *NumberParameter*, or *DateParameter*.

The rule set developer must create a rule attribute on the advice item using the CER Editor and give the rule attribute an appropriate name. The rule set developer must then specify that the rule attribute is of type *StringParameter*, *NumberParameter*, or *DateParameter* as defined in the rule set that is named *CoreAdvisorRuleSet*.

## Creating a link to be displayed in the advice text

To create a link that can be displayed in the advice text, create a link rule class. Then add an attribute to the advice item rule class that is based on the newly created link rule class.

The advisor infrastructure automatically processes all attributes on an advice item that are based on the *AbstractLink* class and convert them to links.

Create a rule class by using the CER Editor and give the rule class an appropriate name. Specify that the rule class inherits from *AbstractLink* defined in the rule set named *CoreAdvisorRuleSet*.

Table 1 describes the mandatory rule attributes that must be specified for each link rule class:

Table 4: Mandatory Rule Attributes

This table describes the mandatory rule attributes that must be specified.

| Rule Attribute Name | Attribute Type | Purpose  |
|---------------------|----------------|--|
| <i>target</i>       | String         | The target of the link. If this is an internal link, the target is the <i>PAGE_ID</i> of the UIM page that this link is to. If the link is an external link, the target is a fully qualified URL.  |
| <i>name</i>         | String         | The name of the link. This name is a logical name that is used to reference the link in the advice text.   |
| <i>modal</i>        | Boolean        | If the derivation of this attribute is set to TRUE, the link appears in a modal. If the derivation of this attribute is set to FALSE, the link appears in the main content pane.   |
| <i>external</i>     | Boolean        | If the derivation of this attribute is set to TRUE, the link is interpreted as a link to an external website outside of the application. If the derivation of this link is set to FALSE, the link is interpreted as an internal link, that is, a link to a UIM page. |

## Creating the advice text

Create the advice text.

### About this task

Create advice text by creating an entry in a localized property file in the **Application Resources** area of application administration. To specify the advice text:

## Procedure

1. Log in to application administration.
2. Within the **Shortcuts** area of the **Administration Workspace** area, select **Intelligent Evidence Gathering**.
3. Select **Application Resources** and create a new resource, which is a property called *var* where *AdviceContextName* is the name of the Advice Context as defined in Application Administration. For example, *AdviceContext.PersonHomeAdvisor*.
4. Create a property entry in the property file `<adviceText>` where *adviceText* is the value of the attribute *adviceText* on the *AdviceItem* rule class that is described in Table 3.2. The value of this property is the localized text to be displayed in the application.

## 1.3 Managing evidence rule objects

When you run the Advisor, the advice text that is specified in [Creating the advice text on page 14](#), is displayed based on an output of the advice rules that are run against configured evidence types.

The design for advice often requires access to the latest evidence data for a case regardless of whether that evidence is activated. Access to the latest active/in-edit evidence is provided by the Active/In-Edit Succession Set Rule Object Converter, which converts active/in-edit evidence data into rule objects for use by CER.

When evidence data changes, the corresponding rule object propagator (Active/In-Edit Succession Set Rule Object Propagator) notifies the Dependency Manager of the change in evidence so that previously calculated advice can be marked as requiring recalculation.

### Active/in-edit succession set rule objects

The active/in-edit Succession set rule object converter automates the conversion of a succession set of evidence to a rule object. Conversion happens regardless of whether the succession set contains active evidence records or the evidence records that are in-edit.

**Important:** Do not confuse the Active/In-Edit Succession Set Rule Object Converter with the Active Succession Set Rule Object Converter. Also do not confuse the Active/In-Edit Succession Set Rule Object Propagator with the Active Succession Set Rule Object Propagator.

However, the active/in-edit succession set rule objects share important characteristics with active succession set, including the population of timeline/non-timeline data and relationships to other rule objects. See the *Inside Cúram Eligibility and Entitlement Using Cúram Express Rules* guide for details on processing for active succession set rule objects.

Evidence functions allow developers to store records of evidence that can change over time. When circumstances change in the real world, a user can record those changes in the system by "succeeding" an earlier version of evidence. These versions of evidence make up a "succession

set" which describe the history of some real-world evidence. Advisor rules treat each piece of changeable evidence as one rule object, which has timeline-based attributes.

The Active/In-Edit Succession Set Rule Object Converter automates the conversion of a succession set of evidence rows into a single rule object with a mixture of timeline and non-timeline attributes. Changes to evidence go through a lifecycle. Transitions between the states of the evidence lifecycle are considered by the Active/In-Edit Succession Set Rule Object Propagator.

The Active/In-Edit Succession Set Rule Object Converter also populates relationships between rule objects for parent and child succession sets, if required.

## Configuration

The converter and propagator share a common set of configuration data, and accept configurations, which adhere to the following structure:

- Propagator type must be *"ROPT2011"* (the code for *'Active/InEdit Succession Set Propagator'* from the *'RuleObjectPropagatorType'* code table).
- Each evidence type to be converted or propagated must be listed in an 'evidence' element with a type exactly matching the evidence's type from the 'EvidenceType' code table.
- Each conversion/propagation target must be listed as a 'ruleset' element (within the 'evidence' element), specifying the name of the rule set to target, and optionally the rule class (if the name of the rule class differs from that of the database table).



Configurations are cumulative. That is, there might be many configurations of type "ROPT2011", and if an evidence type is present in any of those configurations then the evidence type is converted and propagated. Otherwise, the evidence type is ignored.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Code for the 'Active/Edit Succession Set Propagator' from
the 'RuleObjectPropagatorType' code table./>
-->
<propagator type="ROPT2011">
  <configuration>
    <!--
    The code for your evidence type.
    The type must exactly match the code from the
    'EvidenceType' code table including upper/lower case.
    -->
    <evidence type="ET10069">
      <ruleset name="TestActiveInEditPropagatorRuleSet"
ruleclass="SimpleTestActiveInEditPropagatorRuleSet"/>
      <!--
      Map this evidence type to
      TestActiveInEditPropagatorRuleSet.
      SimpleTestActiveInEditPropagatorRuleSetCalculatedDates,
      and also to CGISSAdvisorEvidenceRuleSet.LivingArrange
      -->
      <ruleset name="TestActiveInEditPropagatorRuleSet"
ruleclass=
"SimpleTestActiveInEditPropagatorRuleSetCalculatedDates"/>
      <ruleset name="CGISSAdvisorEvidenceRuleSet"
ruleclass="LivingArrange"/>
    </evidence>
    <evidence type="ET10090">
      <ruleset name="CGISSAdvisorEvidenceRuleSet"
ruleclass="HouseholdRelationship"/>
    </evidence>
    <evidence type="ET10087">
      <ruleset name="CGISSAdvisorEvidenceRuleSet"
ruleclass="Deprivation"/>
    </evidence>
  </configuration>
</propagator>
```

Figure 1: Sample configuration for the Active/Edit Succession Set Rule Object Converter and Propagator

The following types of configuration problems are detected by the Active/Edit Succession Set Rule Object Converter/Propagator processing:

- Evidence type not specified in the evidence element.
- The evidence type with the specified type code could not be found.
- The targeted rule class does not extend the `PropagatorRuleSet.ActiveInEditSuccessionSet` rule class.
- A rule class is targeted by more than one source evidence type.

## Conversion processing

Each evidence type can map to a number of target rule classes, according to the configurations for the Active/Edit Succession Set Rule Object Converter held on the system. However, for the sake of clarity, this section describes the behavior of the Active/Edit Succession Set Rule Object Converter where an evidence type is mapped to a single rule class only.

## Overview

When an Active/In-Edit Succession Set Rule Object is requested during a CER calculation, the Active/In-Edit Succession Set Rule Object Converter is started to populate that rule object. The Object Converter retrieves the latest evidence rows in the succession set and uses them to populate the attribute values on the rule object. The values of the evidence fields are used to map to identically named rule attributes on the rule class. Any evidence field without a corresponding rule attribute is ignored. Evidence fields are defined by:

- **Dynamic evidence.** The evidence fields available are those defined by the dynamic evidence metadata for the evidence type (see the *Cúram Dynamic Evidence Configuration Guide*);
- **Non-dynamic evidence.** The evidence fields available are those defined on the evidence-specific database table that is modeled for the static evidence type (see the *Cúram Evidence Guide* and the *Cúram Evidence Generator* guides).

When populating a particular attribute value on a rule object, the behavior of the Active/In-Edit Succession Set Rule Object Converter differs according to whether the rule attribute's type is a timeline. The Object Converter also contains special processing to populate rule attributes to point to rule objects for related succession sets.

## Timeline-based data types

If the data type of the attribute is *Timeline<some data type>*, then the Active/In-Edit Succession Set Rule Object Converter allows the evidence value to differ (across different evidence versions in the succession set). The converter forms a timeline value with the start date equal to the start of the evidence lifetime, end date equal to the end of the evidence lifetime, and accumulated varying values over the lifetime of the evidence.

## Non-timeline based data types

If the data type of the attribute is not *Timeline<some data type>*, then the converter does not allow the evidence value to differ across different evidence versions in the succession set. Each version of evidence in the succession set bears the same data value for the evidence field, and this single data value is used to populate the rule attribute value.

## Population of relationships to rule objects for other succession sets

When a succession set of active evidence is converted to a rule object, then any rule attributes that are annotated with the *'relatedSuccessionSet'* are automatically populated with rule objects for related succession sets:

- **Parent:** the attribute is populated with the rule objects for the succession sets for the parent(s) of the evidence; or
- **Child:** the attribute is populated with the rule objects for the succession sets for the child/children of the evidence.

The type of the related evidence is identified from the type of the attribute, which can either be a rule class (extending *'ActiveInEditSuccessionSet'*) or a list of such rule classes.

### **Rule attributes inherited from *ActiveInEditSuccessionSet***

Each rule class that is targeted by the Active/In-Edit Succession Set Rule Object Propagator must ultimately extend the '*PropagatorRuleSet.ActiveInEditSuccessionSet*' rule class, and so inherits the following rule attributes:

#### **Rule attributes**

- **successionID**  
Populated from the '*successionID*' value on the '*Evidence Descriptor*' rows, and used to uniquely identify the rule object (among other rule objects of the same rule class).
- **caseID**  
Populated from the '*caseID*' value on the '*EvidenceDescriptor*' row. If the evidence relates to an integrated case, the case ID is that of an integrated case. If the evidence relates to a product delivery case, the case ID is that of the particular product delivery that holds the evidence.
- **description**  
Contains a default rule to derive a description for the succession set rule object. Subclasses can override this description.
- **exists**  
A Boolean timeline that indicates the period for which the succession set rule object "exists" for the dates between the designated start and end dates (inclusive). The Boolean is false for dates before the start of the lifetime or after its end, if any.
- **evidenceDescriptorID**  
A Number timeline, which is populated from the '*evidenceDescriptorID*' value on the '*EvidenceDescriptor*' rows that make up the succession set. The values vary according to the evidence row "in effect" at various points along the lifetime of the succession set rule object. Each value uniquely identifies the active '*EvidenceDescriptor*' row that contains the source of the data in effect on a particular date on the timeline-based attributes on the rule object. These values change when an evidence correction is activated because at that point a different evidence row becomes an active member of the succession set.

#### **Restrictions on access**

Use CER's `<readall>/<match>` expression in your CER rule sets to access rule objects converted from active/in-edit succession set data. You can only specify the `retrievedattribute` to be the `caseID`.

If you specify `retrievedattribute` to be the name of any other attribute, then the Active/In-Edit Succession Set Rule Object Converter throws a runtime exception when the CER `<readall>/<match>` expression is run.

**Tip:** If you require only some of the active evidence row evidence of a given type for a case, then wrap the `<readall>/<match>` expression within a `<filter>` expression. For example, use `<readall>/<match>` matching on `caseID` to find all the Income active/in-edit succession set rule objects for a case. Then, use a `<filter>` to restrict the rule objects to those for a particular member of the case.

You can specify the `ruleset` and `ruleclass` for the `<readall>` expression to be a rule class mapped by the data configuration. If you specify a *ruleclass* that is not directly mapped (for example, a base rule class that you have created from which your concrete rule classes inherit) then no rule objects are found.

**Important:** Do not use a `<readall>` without a `<match>`.  
Such an unqualified `<readall>` would typically retrieve many rule objects and no dependency on the overall set of rule objects is stored.

Precedents identified

Table 5: Precedents Identified for Active/In-Edit Succession Set Rule Objects

| Name                            | When Identified   | Trigger for Recalculation   |
|---------------------------------|---|---|
| Active/In-Edit Evidence         | Identifies any case for which <ul style="list-style-type: none"><li>A search was run to retrieve Active/In-Edit Succession Set Rule Objects; and/or</li><li>One or more attribute values were accessed for one or more Active/In-Edit Succession Set Rule Objects for the case's evidence.</li></ul> The precedent ID refers to the <code>caseID</code> that owns the evidence that was accessed. | If active evidence is edited or canceled, or changes made to in-edit evidence for a case, then a precedent change item for the case is written to a precedent change set.   |
| Rule Object Data Configurations | Identifies the use of the configuration for the Active/In-Edit Succession Set Rule Object Converter if any Active/In-Edit Succession Set Rule Object is accessed during the calculation.  | If changes to the data configuration for the Active/In-Edit Succession Set Rule Object Converter are published, then a precedent change item for the converter's data configuration is written to a precedent change set. |

**Note:** In practice, the two conditions described in the table for the Active/In-Evidence Succession Set Rule Objects amount to the same thing. That is, that Active/In-Edit Succession Set Rule Objects for the case's evidence were accessed in some way. Generally, a search is run to retrieve rule objects so that one or more attribute values can be accessed on those rule objects anyway. Note also that the *activation* of in-edit evidence changes for a case does not cause a precedent change item to be written as the newly-activated evidence is still the latest active/in-edit evidence for the case.

Propagation processing

When evidence edits are made for an evidence type that is configured for Active/In-Edit Succession Set Rule Objects, then the Active/In-Edit Succession Set Rule Object Propagator

listens to internal events from the Evidence Controller, requests the corresponding rule object, and manipulates it in memory.

A rule object can be created, modified, or removed, according to whether evidence is being created, edited, or canceled (in the evidence workspace, before activation).

The Active/In-Edit Succession Set Rule Object Propagator informs the Dependency Manager of evidence edits that have been made so that the Dependency Manager can determine the effects of those changes, in particular to mark advice as requiring recalculation. Dependencies on active/in-edit evidence are stored at the case level, by recording a dependency on the caseID of the case that owns the evidence.

## Example

Let's say that a person's income from an employment is modeled as *Cúram* Evidence. The income starts when a person starts an employment, and ends if the employment is terminated. The name of the employer is constant throughout the period of income because the designer of the evidence structure decided that if a person changes jobs, then the first employment ends and a separate employment starts.

Over the lifetime of an employment, the income amount (that is, the per-annum pay) can vary, as the employee receives pay rises. Similarly, but independently, the person can be employed on a permanent or temporary basis, and this "employment status" can change over the lifetime of the employment. It is possible for the income's amount to change on the same date as the employment status, but a change in income amount can occur without a change in employment status.

The evidence designer designs an income evidence entity as follows:

- *startDate* The date that the income (that is, the overall employment) started
- *endDate* The date that the income (that is, the overall employment) ended, if any.
- *employer* and Identifier of the employer, constant throughout the income (see the design decision that is described)
- *amount* and The per-annum pay amount
- *employmentStatus* Code for whether the employment status is permanent or temporary.

A rules designer then models a new "Income" rule class, extending the "ActiveInEditSuccessionSet" rule class, and adds rule attributes, identifying which have values that change over time (that is, those which are allowed to vary across different records in the same succession set):

Should be constant across records in the succession set:

- *startDate*
- *endDate*
- *employer*

Should be allowed to vary across records in the succession set:

- *amount*
- *employmentStatus*

The rules designer also identifies which rule attributes identify the "lifetime" of the Income:

- startDate
- endDate

The designer annotates the rule class to identify these rule attributes.

An administrator publishes the rule set changes, and then publishes a data configuration for Active/In-Edit Succession Set Rule Objects to map the Income evidence type to the new rule class. A caseworker records some new Income evidence (for an employment that started on 1 January 2000).

Initially the evidence is "in edit" and its data is available to the Active/In-Edit Succession Set Rule Object Converter to populate a rule object. When evidence capture is complete, the case worker activates the evidence. No action is taken by the Active/In-Edit Succession Set Rule Object Propagator.

Over time circumstances change:

- on 1 January 2001, the income amount increases; and
- on 1 May 2002, the employment status changes from "temporary" to "permanent"

On each of these occasions, the case worker records a new version of the Income evidence, leading to the system storing a new 'EvidenceDescriptor'/'Income' pair of rows for the evidence data effective from each change date.

The Active/In-Edit Succession Set Rule Object Converter recognizes that the three versions of evidence relate to a single succession set, and use the effective-dated data to change the timeline values for the attributes on the single rule object. The rule object data is updated as soon as the changes are made, there is no wait until the succession set is activated.

On 30 June 2002, the employment comes to an end and a caseworker records the end date on the latest record in the succession set. The caseworker inserts the changes, which cause the existing latest "EvidenceDescriptor"/"Income" pair to become "superseded" and a new pair to become "active". The Active/In-Edit Succession Set Rule Object Converter immediately updates the rule object to change its timeline values from 1 July 2002 (the day after the end of the employment).

Some time later, a review of the case finds that the entire history of the income that is recorded against the wrong person. All the evidence records for the Income are canceled by the caseworker, pending removal, which causes existing rule object to be removed. The caseworker then realizes that he has canceled the Income record for the wrong person. He reverts the changes and the appropriate rule object gets re-created. The case worker now cancels the Income records for the correct person.

## 1.4 Example Advisor configuration

---

A worked example of how to configure the Advisor to display some advice. The example shows you how to create an Advisor rule set by using the CER Editor, configure the advice context, and specify the advice text.

To fully understand the CER Editor-related section of this appendix, you must be familiar with the CER Editor. If you are not already familiar with the basic concepts of the CER Editor,

read *Working with Curam Express Rules* guide and *Cúram Express Rules Reference Manual*.

## Create an Advisor rule set using the CER editor

Create a rule set using the CER editor to determine whether the advice is displayed.

The example advice is displayed if a person's marital status is "married" and no relationship of type "spouse" has been entered. Therefore, we must configure a rule set that first looks at a person's marital status and then looks at the person's relationships in order to evaluate whether or not to display the advice.

Start by creating the new rule set:

1. Log in to Application Administration.
2. Within the Shortcuts area of the Administration Workspace area, select **Rules and Evidence**.
3. Select the **Cúram Express Rule Sets** link.
4. From the **Action** menu, select **New Rule Set** to create a new rule set.
5. Name the rule set **SpouseRuleSet** and click **Save**. Assign the rule set to a category if appropriate.
6. In the **Cúram Express Rule Sets** tab, search for the newly created **SpouseRuleSet**.
7. In the **Actions** menu for the **SpouseRuleSet**, select **Continue Editing**. A new tab is displayed. The CER editor starts in the new tab.

We now need to design the logic for the **SpouseRuleSet**. From a business perspective, the advice is provided on the **Person** home page under the condition that this person is married and no spousal relationships have been recorded for the person. As this is simple logic, we can jump directly to the CER Editor to design the rules implementation.

The implementation works as follows. First we determine the person for which the advice is displayed using the *concernRoleID* parameter from the *Person\_homePage*. The *concernRoleID* is received from the Advice Context. We then search for the person and his or her associated relationships using the *concernRoleID*. Finally, by checking if the person's marital status is equal to married and comparing the number of spouse relationships with '0', we can determine if the advice should be provided. To design this implementation, we complete the following steps in the CER Editor:

1. From the menu item, create new class called *SpouseAdvice*.
2. In the **Properties** tab, click the **Edit** link beside the **Extends** field.
3. Use the **Change** link to let the *SpouseAdvice* rule class extend *AbstractAdviceContext* within the *CoreAdvisorRuleSet*.
4. From the **Attribute** menu item, create the following new attributes for the *SpouseAdviceRuleClass*: *concernRoleID*, *adviceContextID*, *advice*, and description in the *SpouseAdvice* rule class.
5. Set the Data Type field for each new attribute. For the *concernRoleID* attribute, set the **Data Type** field to be *NumberParameter* from the *CoreAdvisorRuleSet*. For the *adviceContextID* attribute, set the **Data Type** field to be **Number**. For the *advice* attribute, set the **Data Type** field to be List of AbstractAdviceItem in the *CoreAdvisorRuleSet*. For the description attribute, set the Data Type field to be Message. From the Rule Class menu item, create a



new rule class called SpouseAdviceItem. Let the SpouseAdviceItem rule class extend from AbstractAdviceItem within the CoreAdvisorRuleSet.

6. From the Attribute menu item, create the following new attributes for the SpouseAdviceItem rule class: adviceText, showAdvice, priority, status, category, adviceContext, description, and ConcernRoleID.
7. Set the Data Type field for each new attribute. For the adviceText attribute, set the Data Type to be String. For the showAdvice attribute, set the Data Type field to be Boolean. For the priority attribute, set the Data Type field to be the AdvicePriority code table. For the status attribute, set the Data Type field to be the AdviceStatus code table. For the category attribute, set the Data Type field to be the AdviceCategory code table. For the adviceContext attribute, set the Data Type field to be Number. For the description attribute, set the Data Type field to be Message. For the concernRoleID attribute, set the Data Type field to be Number.
8. Drag a Code Table operator onto the priority attribute. Within the Properties tab of the operator, set the Table to be AdvicePriority and set the value to be AP2001.
9. Drag a Code Table operator on the status attribute. Within the Properties tab of the operator, set the Table to be AdviceStatus and set the value to be AS2002.
10. Drag a Code Table operator on the category attribute. Within the Properties tab of the operator, set the Table to be AdviceCategory and set the value to be AC2001.

Now we have extended the essential attributes and we can go on to add the logic to the rules. First we must determine if the person is married or not as follows:

1. Create a new Person attribute for the SpouseAdviceItem rule class. For the Person attribute, set the Data Type field to be Person from the ruleset ParticipantEntitiesRuleSet.
2. Drag a Search operator onto the Person attribute. Double click the Search operator to open the Edit Search dialog, enter the Data Type Person previously mentioned.
3. Add a New Match to the Search operator and then drag a reference to it that references the concernRoleID attribute within the SpouseAdviceItem rule class.
4. Check the Single Item checkbox in the Properties tab of the Search operator.
5. Create a new attribute for the SpouseAdviceItem rule class and call it isMarried.
6. Set the Data Type field for the new attribute to be Boolean.
7. Drag a Compare operator to the new attribute. Edit its expression to compare the maritalStatus within the Person attribute to the Codetable MaritalStatus with the value MS2.

Now we calculate how many spouse relationships the person has as follows:

1. Create a new attribute for the SpouseAdviceItem rule class and call it relationships. For the relationships attribute, set the Data Type field to be a List of type ConcernRoleRelationship from the rule set ParticipantEntitiesRuleSet.
2. Drag a Search operator onto the Relationships attribute. Double click the attribute to open the Edit Search dialog, enter the Data Type ConcernRoleRelationship just mentioned.
3. Add a New Match to the Search operator and drag a reference to it that references the concernRoleID attribute within the SpouseAdviceItem rule class.
4. Create a new attribute for the SpouseAdviceItem rule class and call it numOfSpouseRelationships with the Data Type Number.
5. Drag a Size operator onto the new attribute. Then drag a Filter operator inside the Size operator.



6. Within the Filter operator, for the Empty List element, let this refer to the relationships attribute. For the Empty Member element, drag a new Compare operator onto it that compares the relationshipType in within the relationships attribute with the Codetable RelationshipTypeCode value RT6.

Finally, we can determine the result of the previous steps as follows:

1. Open the showAdvice attribute within the SpouseAdviceItem rule class.
2. Drag the AND operator onto it, making the isMarried the first condition. For the second condition, add a Compare operator beside the isMarried condition that compares if the numOfSpouseRelationship is equal to 0.
3. Open the adviceText attribute within the SpouseAdviceItem rule class.
4. Drag a String properties operator onto the attribute and within the Properties Tab, give it a value of startCapture.
5. Open the advice attribute within the SpouseAdvice rule class.
6. Drag a Fixed List operator onto the attribute, and fill the Data Type of this operator to be AbstractAdviceItem.
7. Drag a Create operator onto the Fixed List operator, and then create the SpouseAdviceItem rule class with parameters. The parameter concernRoleID within SpouseAdviceItem must refer to concernRoleID in SpouseAdvice. The parameter adviceContext within SpouseAdviceItem must refer to adviceContextID in SpouseAdvice.
8. Save and validate the rule set within the CER Editor.
9. If no errors occur on validation, publish the rule set in application administration using the Publish option on the Cúram Express Rule Sets tab.

## Configure the Advice context and Advice text

Configure the advice context in the dynamic environment and assign the newly created rule set to it.

### Advice context

Configure the Advice context as follows:

1. In the **Shortcuts** panel of the **Administration Workspace**, select **User Interface**.
2. Select the **Advisor** link to open the **Advisor list** page.
3. Configure a new advice context and name it **SpouseRules**.
4. Configure a new advice context key for the newly created advice context. Set the name of the advice context key to be **Person\_homePage** and set the **Type** to be **Page**.
5. Select the **Add Rule Set** link the on the **Actions** menu. Select the **SpouseRuleSet** from the drop-down list to assign the rule set to the Spouse Rules advice context.
6. In the **Actions** menu, add the **Page Parameter** under **Person\_homePage**. The Parameter Name is set to **concernRoleID** and **Type** is set to **Number**.

### Advice text

Specify the advice text to be displayed as follows:

1. In the **Shortcuts** panel of the **Administration Workspace**, select **Intelligent Evidence Gathering**.
2. Select the **Application Resources** link to open the **Application Resources list** page.
3. Create a text file and give it an appropriate name, for example *SpouseRules.properties*.
4. Add the property entry "AdviceItem.startCapture=Capture the spouse relationship of this person." to this file.
5. Select **Add Resource**. In the dialog that opens, name the new resource **AdviceContext.SpouseRules** and ensure that the **Content Type** is **text/plain**.
6. Upload the newly created text file.

If the Advisor component has been installed in the application, the advice is displayed in the **Smart** panel on the **Person** home page. The following advice is displayed if the person is married and no spouse relationships have been recorded:

"Capture the spouse relationship of this person".

# Notices

---

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 Merative 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 Merative

## **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 Merative.

## **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.

Merative 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 Merative, 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.

MERATIVE 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.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

Information concerning non-Merative products was obtained from the suppliers of those products, their published announcements or other publicly available sources. Merative has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-Merative products. Questions on the capabilities of non-Merative products should be addressed to the suppliers of those products.

Any references in this information to non-Merative 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 Merative product and use of those websites is at your own risk.

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 actual people or business enterprises is entirely coincidental.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

#### **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 Merative, 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. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

## ***Privacy policy***

---

The Merative privacy policy is available at <https://www.merative.com/privacy>.

## ***Trademarks***

---

Merative™ and the Merative™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

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.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, Windows™, and the Windows™ logo are trademarks of Microsoft™ Corporation in the United States, other countries, or both.

UNIX™ is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.