

# **Cúram 8.2.2**

**Cúram Incremental Modernization and  
Transformation (IMT) Web Services Cookbook**



## Note

---

Before using this information and the product it supports, read the information in [Notices on page 39](#)



# Edition

---

This edition applies to Cúram 8.2.2.

© Merative US L.P. 2012, 2026

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 Cúram Incremental Modernization and Transformation Web Services</b>	
<b>Cookbook.....</b>	<b>9</b>
1.1 Introduction.....	9
Purpose.....	9
Audience.....	9
Prerequisites.....	9
Chapters in this Guide.....	9
1.2 Register Person.....	10
The Register Person Service.....	10
Incoming Parameters.....	10
Response Message.....	14
1.3 Claim Intake.....	15
The Claim Intake Service.....	16
Incoming Parameters.....	16
Response Messages.....	17
1.4 Evidence Maintenance.....	18
Evidence Maintenance Service.....	18
Creating Evidence.....	18
Reading Evidence.....	20
Activating Evidence.....	24
1.5 Verification.....	30
Verifying Evidence.....	30
Verification Result.....	30
Verification Check.....	32
1.6 Determination.....	34
The Determination Service.....	34
Incoming Parameters.....	35
Response Message.....	36
1.7 Triage.....	37
The Triage Service.....	37
Incoming Parameters.....	37
Response Message.....	38
<b>Notices.....</b>	<b>39</b>
Privacy policy.....	40

Trademarks..... 40

# 1 Cúram Incremental Modernization and Transformation Web Services Cookbook

---

The required input and response parameters for Cúram web service are described. Examples of the expected XML messages that are sent and received from Cúram web services are provided. The following web services are available as part of Cúram: register person, claim intake, evidence maintenance, verification, determination, and triage.

## 1.1 Introduction

---

### Purpose

This guide is intended as a reference handbook for developers working on Cúram Web Services. The guide lists the required input and response parameters for each service, and gives examples of the expected XML messages sent and received.

Cúram Web Services are a means of providing services that are normally available within the Cúram system only, to external systems. The web services currently available are:

- Register Person
- Claim Intake
- Evidence Maintenance
- Verification
- Determination
- Triage

### Audience

This guide is intended for developers working on Cúram Web Services.

### Prerequisites

To best make use of this guide, the reader should have experience in developing the functionality which is available in the application.

### Chapters in this Guide

The following list describes the chapters within this guide:

- **Register Person**  
Register Person gathers the information required to create a person in the Cúram system.
- **Claim Intake**  
Claim intake gathers the information required to create a case within the Cúram system.

- **Evidence Maintenance**

Evidence is the data used to determine entitlement for benefits and services. This chapter describes the requirements for creating, reading and activating evidence.

- **Verification**

Verification confirms the accuracy of information given by clients seeking assistance from SEM agencies.

- **Determination**

Determination takes information gathered in the Cúram system as part of intake, and applies it against enterprise-specific and program-specific rules to create eligibility decisions.

- **Triage**

Triage applies an initial level of review to a basic set of information, to determine a client's need or likely benefit from a program or service.

## 1.2 Register Person

### The Register Person Service

Register Person gathers the information required to create a person in the Cúram system. This service equates to the Register Person business process currently available in the Cúram system. In addition it can operate off a list to allow for registration of more than one client in a single web service call. This will optimize performance as it prevents the overhead involved in calling the service multiple times.

When this service is complete, the person details will be stored in the Cúram system for later reference and use by subsequent services. For example, the person record may be referenced by the Claim Intake service, to allow a product delivery case to be created in the Cúram system against a previously registered person. Person details may be used by the Determination web service.

Certain data items are mandatory as part of the Register Person business process currently available in the Cúram system, and therefore must also be populated in the web service. Otherwise the web service will require exception handling.

### Incoming Parameters

#### **Minimum Requirements**

The parameters are used to populate the internal struct: `core.facade.PersonRegistrationDetails`:

Table 1: Fields

Intake Element	Map to Parameter	Schema Type
firstname	firstForeName	xs:string.
surname	surname	xs:string
gender	sex	bt:codetablecode

Intake Element	Map to Parameter	Schema Type
dateOfBirth	dateOfBirth	bt:date
dateOfRegistration	registrationDate	bt:date
maritalStatus	currentMaritalStatus	bt:codetablecode
nationality	nationality	bt:codetablecode
countryOfBirth	countryOfBirth	bt:codetablecode
addressLayout	addressData.addressLayoutType	bt:codetablecode
addressLine1	addressData.addressLayoutType	bt:codetablecode

## Incoming Parameter Descriptions

Table 2: Parameter Descriptions

Parameter	Domain	Description
firstForeName	FIRST_FORENAME	The first name of the person to be registered. Type: string
surname	SURNAME	The surname of the person to be registered. Type: string
sex	GENDER_CODE	The gender of the person to be registered. Code table: Gender
dateOfBirth	CÚRAM_DATE	The date of birth of the person to be registered. Format: ddMMyyyy
registrationDate	CÚRAM_DATE	The date of the persons registration. Format: ddMMyyyy
currentMaritalStatus	MARITAL_STATUS_CODE	The marital status of the person to be registered. Code table: MartialStatus
nationality	NATIONALITY_CODE	The nationality of the person to be registered. Code table: Nationality
birthCountry	COUNTRY_CODE	The country of birth of the person to be registered. Code table: Country
addressData.addressLayoutType	ADDRESS_DATA	The address layout type for the incoming address details. Code table: AddressLayoutType
addressData.addressLine1	ADDRESS_DATA	The first line of the address of the person to be registered. Type: string

## Optional Incoming Parameters

Table 3: Additional Parameters

Intake Element	Map to Parameter	Schema Type
address.addressLine2	addressData.addressLine2	bt:addressdata
address.addressLine3	addressData.addressLine3	bt:addressdata
address.addressLine4	addressData.addressLine4	bt:addressdata
address.addressLine5	addressData.addressLine5	bt:addressdata

Intake Element	Map to Parameter	Schema Type
address.city	addressData.city	bt:string
address.county	addressData.county	bt:codetablecode
address.country	addressData.country	bt:codetablecode
address.postalCode	addressData.postalCode	bt:codetablecode
address.statecode	addressData.statecode	bt:codetablecode
address.comments	addressData.comments	bt:string
address.statusCode	addressData.statusCode	bt:codetablecode
address.zipCode	addressData.zipCode	bt:codetablecode
addressType	addressType	bt:codetablecode
addressIndicator	addressIndicator	bt:boolean
mailingAddress.addressLayout	mailingAddressData.addressLayout	bt:codetablecode
mailingAddress.addressLine1	mailingAddressData.addressLine1	bt:addressdata
mailingAddress.addressLine2	mailingAddressData.addressLine2	bt:addressdata
mailingAddress.addressLine3	mailingAddressData.addressLine3	bt:addressdata
mailingAddress.addressLine4	mailingAddressData.addressLine4	bt:addressdata
mailingAddress.addressLine5	mailingAddressData.addressLine5	bt:addressdata
mailingAddress.city	mailingAddressData.city	bt:string
mailingAddress.county	mailingAddressData.county	bt:codetablecode
mailingAddress.country	mailingAddressData.country	bt:codetablecode
formattedAddress	formattedAddressData	bt:addressData
othername	otherForename	bt:string
type	type	bt:codetablecode
title	title	bt:codetablecode
initials	initials	bt:string
suffix	nameSiffix	bt:string
ssn	socialSecurityNumber	bt:string
motherBirthSurname	motherBirthSurname	bt:string
preferredName	preferredName	bt:string
verifiedDateOfBirth	dateOfBirthVerified	bt:boolean
dateOfDeath	dateOfDeath	bt:date
verifiedDateOfDeath	dateOfDeathVerified	bt:boolean
specialInterest	specialInterest	bt:codetablecode
phoneType	phoneType	bt:string
phoneCountry	phoneCountryCode	bt:codetablecode
phoneAreaCode	phoneAreaCode	bt:int32
phoneNumber	phoneNumber	bt:int32
phoneExtension	phoneExtension	bt:int32
contactPhoneNumber	contactPhoneNumber	bt:int32
contactPhoneCountry	contactPhoneCountryCode	bt:codetablecode

Intake Element	Map to Parameter	Schema Type
contactPhoneArea	contactPhoneAreaCode	bt:int32
contactName	contactName	bt:string
contactPhoneExtension	contactPhoneExtension	bt:int32
contactEmail	contactEmailAddress	bt:string
contactEmailType	contactEmailType	bt:codetablecode
contactTitle	contactTitle	bt:string
publicOffice	publicOfficeID	bt:int16
preferredPOfficeContact	preferredPublicOfficeContact	bt:string
preferredPOfficeName	preferredPublicOfficeName	bt:string
preferredLanguage	preferredLanguage	bt:codetablecode
placeOfBirth	birthPlace	bt:string
concernID	concernID	bt:long
ethnicOrigin	ethnicOriginCode	bt:codetablecode
exceptionMethod	commExceptionMethodCode	bt:codetablecode
exceptionReason	commExceptionReasonCode	bt:codetablecode
exceptionFromDate	commExceptionFromDate	bt:date
exceptionToDate	commExceptionToDate	bt:date
foreignResidencyCountry	foreignResidencyCountryCode	bt:codetablecode
foreignResidencyReason	foreignResidencyReasonCode	bt:codetablecode
foreignResidencyFromDate	foreignResidencyFromDate	bt:date
foreignResidencyToDate	foreignResidencyToDate	bt:date
citizenshipCountry	citizenshipCountryCode	bt:codetablecode
citizenshipReason	citizenshipReasonCode	bt:codetablecode
citizenshipFromDate	citizenshipFromDate	bt:date
citizenshipToDate	citizenshipToDate	bt:date
preferredCommMethod	preCommMethod	bt:codetablecode
relatedClientID	relatedConcernRoleID	bt:clientIdentifier
paymentFrequency	paymentFrequency	bt:frequencyPattern
nextPaymentDate	nextPaymentDate	bt:date
paymentMethod	methodOfPmtCode	bt:string

```
<root>
  <register>
    <registerPerson id="123252">
      <firstname>MARY</firstname>
      <surname>McConnell</surname>
      <gender>SX1</gender>
      <dateOfBirth>06061975</dateOfBirth>
      <dateOfRegistration>12112007</dateOfRegistration>
      <maritalStatus>MS1</maritalStatus>
      <nationality>NT7</nationality>
      <birthCountry>PK</birthCountry>
      <address>
        <addressLayout>US</addressLayout>
        <addressLine1>PineWood</addressLine1>
        <addressLine2>The hills</addressLine2>
        <addressLine3>HillView</addressLine3>
        <addressLine4>The Rise</addressLine4>
        <addressLine5>Malahide</addressLine5>
        <city>Ballymun</city>
        <countryCode>US</countryCode>
      </address>
    </registerPerson>
  </register>
</root>
```

Figure 1: Inbound Example : Register Person

This figure displays an example of the inbound register person xml message.

Response Message

Response Parameters

The parameters are contained within the internal struct: core.facade.PersonRegistrationResult

Table 4: Response Parameters

Map from Parameter	Reponse Element	Type
clientID	clientID	long

## Response Parameter Descriptions

Table 5: Parameter Descriptions

Parameter	Domain	Description
id attribute	n/a	An optional identification, if found in the original inbound details, is included within the response message. This allows the third party to easily match inbound and outbound person registration data. Type: int
clientID	CONCERN_ROLE_ID	The client identifier of the related client. Type: long

```
<receiveDocumentReturn>
  <response>
    <registerPerson id="123252" success="true">
      <clientID>5449355549118300160</clientID>
    </registerPerson>
  </response>
</receiveDocumentReturn>
```

Figure 2: Response Example : Register Person

This figure displays an example of the register person response XML message.

```
<receiveDocumentReturn>
  <response>
    <registerPerson success="false" id="123252">
      <exception>
        <message>An error occurred during the person
          registration process.</message>
        <exceptionMessage>The codetable Gender does not
          contain the value passed: SX100.
        </exceptionMessage>
      </exception>
    </registerPerson>
  </response>
</receiveDocumentReturn>
```

Figure 3: Error Response Example : Register Person

This figure displays an example of the register person error response XML message.

## 1.3 Claim Intake

## The Claim Intake Service

Claim intake gathers the information required to create a case within the Cúram system. This service equates to the Create Product Delivery business process and therefore we do not include details of the creation of Integrated Case, Service Plan or any other case type. The objective is to make a product delivery case available for the subsequent storage of evidence and execution of business services, such as verification and determination.

Certain configuration data is required in order for a product delivery case to be successfully created. This configuration data must be created in advance of using Claim Intake services. Therefore it is assumed that the product and the product provider configuration data are available for this service. Also it is assumed that primary client for whom the claim is being captured has already been registered on the Cúram system using the Register Person service.

## Incoming Parameters

### Minimum Requirements

The parameters are used to populate the internal struct: `core.facade.CreateCaseDetails`

Table 6: Minimum Requirements

Intake Element	Map to Parameter	Schema Type
clientID	clientID	bt:clientIdentifier
productID	productID	bt:productProviderIdentifier
providerID	productProviderID	bt:productProviderIdentifier
providerLocation	providerLocation	bt:providerLocation
deliveryPattern	productDeliveryPatternID	bt:providerDeliveryPatternIdentifier
receivedDate	receivedDate	bt:date
currency	currencyType	bt:codetablecode

### Incoming Parameter Descriptions

Table 7: Parameter Descriptions

Parameter	Domain	Description
clientID	CONCERN_ROLE_ID	The client's identification. Type: long.
productID	PRODUCT_ID	The case product's identification. Type: long.
providerID	PRODUCT_PROVIDER_ID	The case's product provider's identification. Type: long.
providerLocation	PROVIDER_LOCATION	The case's provider's location. Type: int.
deliveryPattern	PRODUCT_DELIVERY_PATTERN_ID	The Identification of the product's delivery pattern. Type: long.
receivedDate	Cúram_DATE	The date of receipt. Format: ddMMyyyy

Parameter	Domain	Description
currency	CURRENCY_CODE	The currency type. Code table: Currency

### Optional Incoming Parameters

Table 8: Additional Parameters

Intake Element	Map to Parameter	Schema Type
objective	objectiveCode	bt:string
caseStartDate	caseStartDate	bt:date

```

<root>
  <claimIntake>
    <claimIntakeGroup>
      <clientID>8232580118833266688</clientID>
      <productID>2111</productID>
      <providerID>123</providerID>
      <providerLocation>2701</providerLocation>
      <deliveryPattern>111</deliveryPattern>
      <receivedDate> 13112007 </receivedDate>
      <currency> USD</currency>
    </claimIntakeGroup>
  </claimIntake>
</root>

```

Figure 4: Inbound Example : Claim Intake

This figure displays an example of the inbound Claim Intake xml message.

## Response Messages

### Response Parameters

The parameters are contained within the internal struct: core.facade.CreatedCaseIDKey

Table 9: Response Parameters

Map from Parameter	Reponse Element	Type
caseID	caseID	long
clientID	clientID	long

### Response Parameter Descriptions

Table 10: Parameter Descriptions

Parameter	Domain/Attribute	Description
caseID	CASE_ID	The case identifier of the created Product Delivery. Type: long

Parameter	Domain/Attribute	Description
clientID	CONCERN_ROLE_ID	The client identifier of the related client. Type: long

```

<receiveDocumentReturn>
  <response>
    <claimIntake success="true">
      <caseID>3278620528725721088</caseID>
      <clientID>-5728578726015270912</clientID>
    </claimIntake>
  </response>
</receiveDocumentReturn>

```

Figure 5: Response Example : Claim Intake

This figure displays an example of the response XML message.

```

<receiveDocumentReturn>
  <response>
    <claimIntake success="false">
      <exception>
        <message>An error occurred during the claim intake
          procedure for the client .</message>
        <exceptionMessage>Record not found.</exceptionMessage>
      </exception>
    </claimIntake>
  </response>
</receiveDocumentReturn>

```

Figure 6: Error Response Example : Claim Intake

This figure displays an example of the error response xml message.

## 1.4 Evidence Maintenance

### Evidence Maintenance Service

Evidence is the data used to determine entitlement for benefits and services. Therefore the presence of this data is required to support other Cúram services in the entitlement area, for example Triage and Determination. Services offered for evidence maintenance are simple in nature. They assume that any approvals are performed in advance by the calling system, and the management of evidence relationships, evidence hierarchies and so on is dealt with in the calling system.

### Creating Evidence

## Incoming Parameters

The Create Evidence service equates to the generic Insert Evidence pattern for case evidence in the evidence framework. In addition, it can operate off a list to allow for insertion of multiple evidence records in a single service call.

Following creation of the evidence records in the Cúram system, they will be activated immediately without initiating evidence approvals. If the activation cannot successfully complete, for example if the evidence must be verified, then the evidence record will be left in an in-edit status. In-edit evidence records can be verified at a later date using the Activate Evidence service when any issues are resolved and the evidence is verified.

The parameters are used to populate the internal struct:

Cúram.core.sl.infrastructure.struct.EvidenceDescriptionInsertDtIs

Table 11: Minimum Requirements

Intake Element	Map to Parameter	Schema Type
caseID	caseID	sem:caseIdentifier
evidenceType	evidenceType	sem:codetablecode
receivedDate	receivedDate	sem:date
effectiveDate	effectiveDate	sem:date
dataObjects	see below	see below

Each Evidence Create schema has an object structure defined for the incoming data. The dataObjects structure is:

```
<dataItem name="{data item name}"
  >{value}</dataItem>
```

- Data Item name: The name of the attribute within the struct that is passed to the entity object.
- Value: The value to populate the struct field with. This will be passed to the entity object.

**Note:** DataItem to struct mapping controls all data type conversions and checks.

## Incoming Parameter Descriptions

Table 12: Parameter Descriptions

Parameter	Domain	Description
caseID	CASE_ID	The numeric identification of the evidence records related case. Type: long
evidenceType	EVIDENCE_TYPE	The evidence type of the evidence record created. Codetable: evidenceType
receivedDate	CÚRAM_DATE	The date of receipt of the evidence creation. Format ddMMyyyy

Parameter	Domain	Description
effectiveDate	CÚRAM_DATE	The date from when the created evidence is effective. Format ddMMyyyy

### Optional Incoming Parameters

Table 13: Additional Parameters

Intake Element	Map to Parameter	Schema Type
participantID	participantID	sem:participantIdentifier

The following figure displays an example of the inbound Create Evidence xml message:

```

<root>
  <evidence>
    <evidenceData>
      <evidenceDetails>
        <caseID>8034421735228964864</caseID>
        <evidenceType>ET500</evidenceType>
        <receivedDate>01010001</receivedDate>
        <effectiveDate>01010001</effectiveDate>
      </evidenceDetails>
      <dataObjects>
        <dataItem name="sportingActivityID">
          -1333065489701666816</dataItem>
        <dataItem name="caseParticipantRoleID">
          -900719925474099200</dataItem>
        <dataItem name="sportingActivityType">SA5</dataItem>
        <dataItem name="sportingAwardType">SAT2</dataItem>
        <dataItem name="paymentAmount">100.00</dataItem>
        <dataItem name="comments"/>
        <dataItem name="startDate">01010001</dataItem>
        <dataItem name="endDate">01010001</dataItem>
        <dataItem name="versionNo">1</dataItem>
      </dataObjects>
    </evidenceData>
  </evidence>
</root>

```

Figure 7: Inbound Example : Create Evidence

## Reading Evidence

### Incoming Parameters

The Read Evidence service equates to the generic View Evidence pattern for case evidence in the evidence framework. In addition once again it can operate off a list to allow for retrieval of more than one evidence record in a single service call. The service is simply to retrieve evidence data from the Cúram system.

The parameters with both Minimum requirements tables are used to populate the internal struct: Cúram.core.sl.infrastructure.struct.EIEvidenceKey

Table 14: Minimum Requirements: Option A

Intake Element	Map to Parameter	Schema Type
evidenceID	evidenceID	sem:evidenceIdentifier
evidenceType	evidenceType	sem:codetablecode

Table 15: Minimum Requirements: Option B

Intake Element	Map to Parameter	Schema Type
evidenceDescriptorID	evidenceDescriptorID	sem:evidenceDescriptorIdentifier

## Incoming Parameter Descriptions

Table 16: Parameter Descriptions

Parameter	Domain	Description
evidenceID	EVIDENCE_ID	The evidence identification of the evidence record to be read. Type: long
evidenceType	EVIDENCE_TYPE	The evidence type of the evidence record to be read. Codetable: evidenceType
evidenceDescriptorID	EVIDENCE_DESCRIPTOR_ID	The evidence descriptor identification of the evidence record to be read. Type: long

The following figure displays an example of the inbound Read Evidence xml message.

```
<root>
  <evidence>
    <evidenceRead>
      <evidenceID>-1333065489701666816</evidenceID>
      <evidenceType>ET500</evidenceType>
    </evidenceRead>
    <evidenceRead>
      <evidenceID>-7259802599321239552</evidenceID>
      <evidenceType>ET500</evidenceType>
    </evidenceRead>
  </evidence>
</root>
```

Figure 8: Inbound Example : Read Evidence

## Response Message

Table 17: Response Parameters

Map from Parameter	Reponse Element	Type
caseID	caseID	long

Map from Parameter	Reponse Element	Type
evidenceType	evidenceType	string
evidenceDescriptorID	evidenceDescriptorID	long
effectiveDate	effectiveDate	date
dataObjects	See below	See below

Each Evidence Read response has a dataObjects element made up of dataItem child elements. The dataItem structure is:

```
<dataItem name="{data item name}"
  >{value}</dataItem>
```

- Data Item name: The name of the attribute within the struct that is passed to the entity object..
- Type: The type of field that will be populated.
- The value to populate the struct field with. This will be passed to the entity object

Type	Description
string	A string.class type value.
Int	An int.class type value.
Short	A short.class type value.
Double	A double.class type value.
Float	A float.class type value.
Long	A long.class type value.
Date	A Cúram.util.type.Date.class type value.
Date	A Cúram.util.type.Date.class type value, format: ddMMyyyy.
DateTime	A Cúram.util.type.DateTime.class type value, format: ddMMyyyy hh:mm:ss.
Money	A Cúram.util.type.Money.class type value.
FrequencyPattern	A Cúram.util.type.FrequencyPattern.class type value.

## Response Parameter Descriptions

Table 18: Parameter Descriptions

Parameter	Domain/Attribute	Description
caseID	CASE_ID	The case identifier for the case to which the evidence record belongs. Type: long
evidenceType	EVIDENCE_TYPE	The evidence type code for the evidence record created. Codetable: EvidenceType
evidenceDescriptorID	EVIDENCE_DESCRIPTOR_ID	The evidence descriptor identifier for the evidence record created. Type: long
effectiveDate	CÚRAM_DATE	The date from when the created evidence is effective. Format ddMMyyyy

The following figure displays an example of the Read Evidence response xml message:

```

<receiveDocumentReturn>
  <response>
    <evidenceRead success="true">
      <caseID>8034421735228964864</caseID>
      <evidenceType>ET500</evidenceType>
      <evidenceDescriptorID>
        684547143360315392
      </evidenceDescriptorID>
      <effectiveDate>11092007</effectiveDate>
      <dataObjects>
        <dataItem name="sportingActivityID" type="long">
          -1333065489701666816</dataItem>
        <dataItem name="caseParticipantRoleID" type="long">
          -900719925474099200</dataItem>
        <dataItem name="sportingActivityType"
          type="string">SA5</dataItem>
        <dataItem name="sportingAwardType"
          type="string">SAT2</dataItem>
        <dataItem name="paymentAmount"
          type="money">100.00</dataItem>
        <dataItem name="comments" type="string"/>
        <dataItem name="startDate" type="date">
          11022007</dataItem>
        <dataItem name="endDate" type="date">
          10092008</dataItem>
        <dataItem name="versionNo" type="int">1</dataItem>
      </dataObjects>
    </evidenceRead>
  </response>
</receiveDocumentReturn>

```

Figure 9: Response Example : Read Evidence

```

<receiveDocumentReturn>
  <response>
    <evidenceRead success="false">
      <exception>
        <message>The evidence read operation failed.
          Please contact the administrator.</message>
        <exceptionMessage>
          Index: 0, Size: 0
        </exceptionMessage>
      </exception>
    </evidenceRead>
  </response>
</receiveDocumentReturn>

```

Figure 10: Error Response Example : Read Evidence

This figure displays an example of the Read Evidence error response xml message.

## Activating Evidence

The Activate Evidence service equates to the generic Apply Changes evidence pattern for case evidence. Like the other evidence services however, it can take a list of evidence descriptor IDs as input and activate each one.

### ***ActivateForUsers***

#### Incoming Parameters

The parameters are used to populate the internal struct: Cúram.core.struct.CaseKey.

Table 19: Minimum Parameter Requirements

Intake Element	Map to Parameter	Schema Type
caseID	caseID	sem:caseIdentifier

#### Incoming Parameter Descriptions

Table 20: Parameter descriptions

Parameter	Domain	Description
caseID	CASE_ID	The case identifier of the case to which the evidence to be activated is related. Type: long

```
<root>
  <evidence>
    <activateForUsers>
      <caseID>8034421735228964864</caseID>
    </activateForUsers>
    <activateForUsers>
      <caseID>8435421537284500864</caseID>
    </activateForUsers>
  </evidence>
</root>
```

Figure 11: Inbound Example : ActivateForUsers

This figure displays an example of the inbound ActivateForUsers xml message.

#### Response Message

Table 21: Response Parameters

Map from Parameter	Response Element	Type
caseID	caseID	long
activated	activated	boolean

## Response Parameter Descriptions

Parameter	Domain/Attribute	Description
caseID	CASE_ID	The case identifier of the case for which all evidence records were to be activated. Type: long
activated	n/a	The indicator as to whether the activation attempt was a success or failure. Type boolean

```
<receiveDocumentReturn>
  <response>
    <activateUserChanges success="true">
      <caseID>-6737385042546262016</caseID>
      <activated>true</activated>
    </activateUserChanges>
  </response>
</receiveDocumentReturn>
```

Figure 12: Response Example : ActivateForUsers

This figure displays an example of the ActivateForUsers response xml message.

```
<receiveDocumentReturn>
  <response>
    <activateAllChanges success="false">
      <caseID>77912273553500958080</caseID>
      <activated>false</activated>
      <exception>
        <message>The evidence activate all changes operation
          failed.Please contact the administrator.
        </message>
        <exceptionMessage>For input string:
          "77912273553500958080"</exceptionMessage>
      </exception>
    </activateAllChanges>
  </response>
</receiveDocumentReturn>
```

Figure 13: Error Response Example : ActivateForUsers

This figure displays an example of the ActivateForUsers error response xml message.

## ActivateChanges

### Incoming Parameters

The parameters are used to populate the internal struct:

Cúram.core.sl.infrastructure.struct.EvidenceEvidenceDescriptionInsertDtIs

Table 22: Minimum Parameter Requirements: Option A

Intake Element	Map to Parameter	Schema Type
caseID	caseID	sem:caseIdentifier
evidenceID	evidenceID	sem:evidenceIdentifier
evidenceType	evidenceType	sem:evidenceType

The parameters are used to populate the internal struct:  
Cúram.core.sl.infrastructure.struct.EvidenceEvidenceDescriptionInsertDtls

Table 23: Minimum Parameter Requirements: Option B

Intake Element	Map to Parameter	Schema Type
caseID	caseID	sem:caseIdentifier
evidenceDescriptorID	evidenceDescriptorID	sem:evidenceDescriptorIdentifier

Incoming Parameter descriptions

Table 24: Parameter Descriptions

Parameter	Domain	Description
caseID	CASE_ID	The case identifier of the case to which the evidence to be activated is related. Type: long
evidenceID	EVIDENCE_ID	The evidence identifier of the evidence record to be activated. Type: long
evidenceType	EVIDENCE_TYPE	The evidence type code of the evidence record to be activated. Codetable: EvidenceType
evidenceDescriptorID	EVIDENCE_DESCRIPTOR_ID	The evidence descriptor identifier of the evidence record to be activated. Type: long

```
<root>
  <evidence>
    <activateChanges>
      <caseID>8034421735228964864</caseID>
      <evidenceID>501</evidenceID>
      <evidenceType >PET10</evidenceType>
    </activateChanges >
  </evidence>
</root>
```

Figure 14: Inbound Example : ActivateChanges: Option A

This figure displays an example of the inbound ActivateChanges xml message.

```

<root>
  <evidence>
    <activateChanges>
      <caseID>8034421735228964864</caseID>
      <evidenceDescriptorID>6719370644036780032
      </evidenceDescriptorID>
    </activateChanges >
  </evidence>
</root>

```

Figure 15: Inbound Example : ActivateChanges: Option B

This figure displays an example of the inbound ActivateChanges xml message.

## Response Message

Table 25: Response Parameters

Map from Parameter	Reponse Element	Type
caseID	caseID	long
evidenceDescriptorID	evidenceDescriptorID	long
activated	activated	boolean

## Response Parameter Descriptions

Table 26: Parameter Descriptions

Parameter	Domain/Attribute	Description
caseID	CASE_ID	The case identifier of the case to which the evidence record activated is related. Type: long
evidenceDescriptorID	EVIDENCE_DESCRIPTOR_ID	The evidence descriptor identifier of the evidence record activated. Type: long
activated	n/a	The Boolean indicator as to whether the activation attempt was a success or failure. Type: boolean

```

<receiveDocumentReturn>
  <response>
    <activateChanges success="true">
      <caseID>7791227355350958080</caseID>
      <evidenceDescriptorID>
        4332462841530417152
      </evidenceDescriptorID>
      <activated>true</activated>
    </activateChanges>
  </response>
</receiveDocumentReturn>

```

Figure 16: Response Example : ActivateChanges

This figure displays an example of the ActivateChanges response xml message.

```
<receiveDocumentReturn>
  <response>
    <activateChanges success="false">
      <caseID>7791227355350958080</caseID>
      <activated>>false</activated>
      <exception>
        <message>The evidence activate changes operation
          failed. Please contact the administrator.</message>
        <exceptionMessage>Record not found.</exceptionMessage>
      </exception>
    </activateChanges>
  </response>
</receiveDocumentReturn>
```

Figure 17: Error Response Example : ActivateChanges

This figure displays an example of the ActivateChanges error response xml message.

ActivateAllChanges

Incoming Parameters

Table 27: Minimum Parameter Requirements

Intake Element	Map to Parameter	Schema Type
caseID	caseID	sem:caseIdentifier

Incoming Parameter Descriptions

Table 28: Parameter Descriptions

Parameter	Domain	Description
caseID	CASE_ID	The case identifier of the case to which the evidence to be activated is related. Type: long

```
<root>
  <evidence>
    <activateAllChanges>
      <caseID>8034421735228964864</caseID>
    </activateAllChanges>
    <activateAllChanges>
      <caseID>8435421537284500864</caseID>
    </activateAllChanges>
  </evidence>
</root>
```

Figure 18: Inbound Example : ActivateAllChanges

This figure displays an example of the inbound ActivateAllChanges xml message.

## Response Message

Table 29: Response Parameters

Map from Parameter	Reponse Element	Type
caseID	caseID	long
activated	activated	boolean

## Response Parameter Descriptions

Table 30: Parameter Descriptions

Parameter	Domain/Attribute	Description
caseID	CASE_ID	The case identifier of the case for which all evidence records were to be activated. Type: long
activated	n/a	The Boolean indicator as to whether the activation attempt was a success or failure. Type: boolean

```
<receiveDocumentReturn>
  <response>
    <activateAllChanges success="true">
      <caseID>7791227355350958080</caseID>
      <activated>true</activated>
    </activateAllChanges>
  </response>
</receiveDocumentReturn>
```

Figure 19: Response Example : ActivateAllChanges

This figure displays an example of the ActivateAllChanges response xml message.

```
<receiveDocumentReturn>
  <response>
    <activateAllChanges success="false">
      <caseID>77912273553500958080</caseID>
      <activated>false</activated>
      <exception>
        <message>The evidence activate all changes operation
          failed.Please contact the administrator.
        </message>
        <exceptionMessage>For input string:
          "77912273553500958080"</exceptionMessage>
      </exception>
    </activateAllChanges>
  </response>
</receiveDocumentReturn>
```

Figure 20: Error Response Example : ActivateAllChanges

This figure displays an example of the ActivateAllChanges error response xml message.

## 1.5 Verification

---

### Verifying Evidence

Verification is the process of confirming the accuracy of information that is given by clients who are seeking assistance from Social Enterprise Management (SEM) agencies.

#### Before you begin

Before you complete the following procedure, you must add a verification to an integrated case.

#### About this task

You can use two services to verify evidence: Verification Result and Verification Check.

The Verification Result service returns the current status of a list of evidence records. Use Verification Result as a single service that operates from a list to verify the status of multiple evidence records.

The Verification Check service checks for outstanding verifications on a case. In Cúram, when there are outstanding verifications on a case, the system prevents the case from moving to the delivery of payments. Therefore, use the Verification Check service to do a similar validation from a third-party system that uses Cúram to capture and verify evidence.

The following assumptions apply to both verification web services:

- The Cúram Verification administration component is installed.
- The verification requirements are defined in the administration component.
- The Cúram system contains the evidence to verify.

#### Procedure

##### Using the Verification Result service to view the status of an evidence record

1. Log on to Cúram as a caseworker.
2. Open an integrated case.
3. Click the **Evidence** tab.
4. In the menu, click **Verifications**.
5. In the **Verifications** page, click the **All** tab.

The status, for example, Not Verified, of the verification item is displayed under the Status heading.

### Verification Result

## Incoming Parameters

The verification result service is used to return the current verification status of a given list of evidence records.

The parameters are used to populate the struct: EvidenceDescriptorKey

Table 31: Minimum Requirements

Intake Element	Map to Parameter	Schema Type
evidenceDescriptorID	evidenceDescriptorID	bt:evidenceDescriptorIdentifier

## Incoming Parameter Descriptions

Table 32: Parameter Descriptions

Parameter	Domain	Description
evidenceDescriptorID	EVIDENCE_DESCRIPTOR_ID	The evidence descriptor identifier for the evidence record for which the verification results are being queried. Type: long

```
<root>
  <verification>
    <verificationResult>
      <evidenceDescriptorID>
        810647932926689280
      </evidenceDescriptorID>
      <evidenceDescriptorID>
        6719370644036780032
      </evidenceDescriptorID>
    </verificationResult>
  </verification>
</root>
```

Figure 21: Inbound Example : VerificationResult

This figure displays an example of the inbound VerificationResult xml message.

## Response Message

The parameters are contained within the struct:

Cúram.verification.sl.infrastructure.struct.EvidenceVerificationDetails

Table 33: Response Elements

Map from parameter	Reponse element	Type
evidenceDescriptorID	evidenceDescriptorID	long
verificationStatus	verificationStatus	boolean

Response Parameter Descriptions

Table 34: Parameter Descriptions

Parameter	Domain/Attribute	Description
evidenceDescriptorID	EVIDENCE_DESCRIPTOR_ID	The evidence descriptor identifier of the evidence record. Type: long
verificationStatus	EVIDENCE_STATUS_CODE	The code table value to show the verification status of the evidence record. Codetable: EvidenceStatus



Figure 22: Response Example : VerificationResult

This figure displays an example of the VerificationResult response xml message.

Verification Check

Incoming Parameters

The verification check service is used to return a list of outstanding evidence records, for a given case..

The parameters are used to populate the struct: EvidenceDescriptorKey.

Table 35: Minimum Requirements

Intake Element	Map to Parameter	Schema Type
caseID	caseID	bt:caseIdentifier

## Incoming Parameter Descriptions

Table 36: Parameter Descriptions

Parameter	Domain	Description
caseID	CASE_ID Long	The case identifier for the case which will be queried for all its evidence and their current verification status.

```
<root>
  <verification>
    <verificationCheck>
      <evidenceDescriptorID>810647932926689280
    </evidenceDescriptorID>
      <evidenceDescriptorID>6719370644036780032
    </evidenceDescriptorID>
    </verificationCheck>
  </verification>
</root>
```

Figure 23: Inbound Example : VerificationCheck

This figure displays an example of the inbound VerificationCheck xml message.

## Response Message

The parameters are contained within the struct:

verification.sl.infrastructure.struct.EvidenceVerificationDetails

Table 37: Response Elements

Map from parameter	Reponse element	Type
caseID	caseID	long
verificationRequired	verificationRequired	boolean
evidenceDescriptorID	evidenceDescriptorID	long

## Response parameter descriptions

Table 38: Parameter Descriptions

Parameter	Domain/Attribute	Description
caseID	CASE_ID	The case identifier for the case to which all the evidence records are connected to. Type: long.
verificationRequired	n/a	A value to state whether the case has any evidence records with outstanding verifications, ie "Not verified". Type: boolean.

Parameter	Domain/Attribute	Description
evidenceDescriptorID	EVIDENCE_DESCRIPTOR_ID	The evidence descriptor identifier of any evidence records that were found to be in a state of "Not verified". Type: long.

```

<receiveDocumentReturn>
  <response>
    <verificationCheck success="true">
      <caseID>-4377498837804122112</caseID>
      <verificationRequired>true</verificationRequired>
      <evidenceDescriptorID>
        810647932926689280
      </evidenceDescriptorID>
      <evidenceDescriptorID>
        6719370644036780032
      </evidenceDescriptorID>
    </verificationCheck>
  </response>
</receiveDocumentReturn>

```

Figure 24: Response Example : VerificationCheck

This figure displays an example of the VerificationCheck response xml message.

## 1.6 Determination

### The Determination Service

Determination takes information that is gathered in the Cúram system as part of intake, and applies it against enterprise-specific and program-specific rules to create eligibility decisions. Determination is different for each program. It requires a ruleset that is defined to evaluate a participant's eligibility for benefit from a particular product.

Determination is a full eligibility test. It requires a full set of business rules and the set of data on which these rules operate. The outcome of running these business rules is that the client is either submitted for eligibility testing or failed during the submittal process.

At minimum, the following information is needed for determination.

- Configuration data that covers the product and its associated set of business rules to define eligibility for this product must exist in the Cúram system.
- The participant must exist in the Cúram system. This can be achieved by using the Register Person service.
- The case must exist in the Cúram system. This can be achieved by using the Claim Intake service.
- All evidence that is used in determination must exist in the Cúram system, and be available for the case. This can be achieved by using a combination of the Evidence maintenance services.

- The decisions that are created by determination are stored in the Cúram system, but only the submitted for approval verification is returned to the calling service.

## Incoming Parameters

A request for determination must include the case identifier which is used to identify the case to be submitted for approval via the populating of the struct core.SubmitForApprovalKey.

The fromDate and toDate are used to determine if the case's current certification date values are currently before the fromDate or after the toDate. If not the case the certification records are modified to incorporate the inbound dates.

Table 39: Incoming Parameters

Intake Element	Map to Parameter	Schema Type
caseID	caseID	bt:caseIdentifier
fromDate	ProductDeliveryCertDiaryDtIs.periodFromDt	FromDt
toDate	ProductDeliveryCertDiaryDtIs.periodToDt	FromDt

## Incoming Parameter Descriptions

Table 40: Parameter Descriptions

Parameter	Domain	Description
caseID	CASE_ID	The case's identification. Type: long
fromDate	CÚRAM_DATE	The date from which the determination period is submitted for the case. Format: ddMMyyyy
toDate	CÚRAM_DATE	The date to which the determination period is submitted for the case. Format: ddMMyyyy

```
<root>
  <determination>
    <submitforapproval>
      <caseID>-4377498837804122112</caseID>
      <fromDate>23112007</fromDate>
      <toDate>30112007</toDate>
    </submitforapproval>
  </determination>
</root>
```

Figure 25: Inbound Example : Determination

This figure displays an example of the inbound Determination xml message.

# Response Message

Table 41: Response Parameters

Map from Parameter	Reponse Element	Type
caseID	caseID	long
submitted	submitted	boolean

## Response Parameter Descriptions

Table 42: Parameter Descriptions

Parameter	Domain/Attribute	Description
caseID	CASE_ID	The case identifier of the case was submitted for aproval. Type: long
submitted	n/a	The indicator as to whether the determination was submitted for approval. Type boolean

```
<receiveDocumentReturn>
  <response>
    <submittedForApproval success="true">
      <caseID> -4998995586381250560 </caseID>
      <submitted> true </submitted>
    </submittedForApproval>
  </response>
</receiveDocumentReturn>
```

Figure 26: Response Example : Determination

This figure displays an example of the Determination response xml message.

```
<receiveDocumentReturn>
  <response>
    <submittedForApproval success="false">
      <caseID> -4998975586381250560 </caseID>
      <exception>
        <message>The determination submit procedure failed.
          Please contact the administrator.
        </message>
        <exceptionMessage>
          Record not found.
        </exceptionMessage>
      </exception>
    </submittedForApproval>
  </response>
</receiveDocumentReturn>
```

Figure 27: Error Response Example : Determination

This figure displays an example of the Determination error response xml message.

## 1.7 Triage

### The Triage Service

Triage applies an initial level of review to a basic set of information, to determine a client's need or likely benefit from a program or service.

Triage is not a full eligibility test. Triage requires a small set of business rules and consequently, a small amount of information to process the rules. The results provided by Triage are indicative, not final they are suggestive of what a client may be entitled to.

Triage is different for each program and therefore this service must be tailored to each solution. The Triage web service provides an easy means of routing to the solution specific service. Each solution requires a ruleset to be executed which evaluates the indicative entitlement for their product.

There a number of requirements that must be met before the Triage service can be successfully run:

- The ruleset must be configured and available in the Cúram system.
- The evidence over which the ruleset runs must exist in the Cúram system. This can be achieved using a combination of the Evidence maintenance services.
- The participant must exist in the Cúram system. This can be achieved using the Register Person service.

### Incoming Parameters

The parameters are used to populate the struct: Cúram.core.sl.struct.CaseIDandTriageTypeKey

Table 43: Minimum Requirements

Intake Element	Map to Parameter	Schema Type
caseID	caseID	bt:caseIdentifier
trriageType	trriageType	bt:trriageType

### Incoming Parameter descriptions

Table 44: Parameter Descriptions

Parameter	Domain	Description
caseID	CASE_ID	The evidence descriptor identifier of the evidence record. Type: long
trriageType	TRIAGE_TYPE_CODE	The code table value to show the type of triage operation to run. Codetable: triageType

## Response Message

The parameters are contained within the struct: core.sl.struct.TriageResult

Table 45: Response Parameters

Map from parameter	Reponse element	Type
caseID	caseID	long
qualified	qualified	boolean
amount	amount	double

### Response Parameter Descriptions

Table 46: Parameter Descriptions

Parameter	Domain/Attribute	Description
caseID	CASE_ID	The case identifier for the case to which the triage relates. Type: long
qualified	n/a	The indicator as to whether the case has qualified for a payment, via the triage process. Type: boolean
amount	CÚRAM_MONEY	The monetary amount due in payment to the case. Type: double

# 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.