Merative™ SPM Platform

Guidelines for Archiving Dependency Data

Merative SPM v6.0.2 EP13 and 6.0.4.0 and later

Author: Clodagh Hannon
Owner: Merative SPM Platform Group

Dated 27 September 2013 - Version 1.0 Draft 1

Document Control Information

Revision

Status: 1.0

Document Author: Clodagh Hannon Document

Owner: Merative SPM Platform Group

Summary of Changes

To request a change to this document, contact the Document Author or Owner. The Document Author or Owner typically allows appropriate designees to update this document.

Changes to this document are summarized in the following table in reverse chronological order (latest version first).

Revision	Date	Created by	Short Description of Changes
1	[27/09/2013]	Clodagh Hannon	Initial Version

Contents

1 Introduction

The purpose of this document is to provide high level guidance to assist customers on archiving Dependency records from the application. The document will briefly describe the Dependency Manager, the various dependent types that can be stored in the Dependency table, and provide guidance on how to retrieve entries for each of these types where the associated case is to be archived.

1.1 Scope

A fully tested archiving process is not provided with Merative Social Program Management. As such, the purpose of this document is to provide guidance on a best effort basis to assist a customer implementing such a strategy.

This document only describes the table released as part of the Merative Social Program Management product v6.0.2 EP13 and v6.0.4.0. It does **not** describe any customisations made by the customer that would have to be taken into account when archiving such data.

Any data archiving should be first completed on a test system and a complete suite of test cases supplied before being applied to a live system.

This document assumes that a list of cases to be archived has already been established, and provides guidance in retrieving the dependency records associated with these cases.

2 Dependency Manager Overview

(Note: For a full description of the Dependency Manager precedent change sets see the guide CER Reference Manual).

When the value of a data item is derived from the value of another, then it is said to be **dependent** on that item, which is referred to as a **precedent**. A dependent data item can depend on one or more precedents. The Dependency Manager is responsible for storing and managing these dependencies.

CER and its clients (such as the Eligibility and Entitlement Engine, and Advisor) integrate tightly with the Dependency Manager to support the recalculation of CER results whenever inputs which have been used in CER calculations change.

The Dependency table stores the details of the reliance of a dependent on a precedent. If the system has a reassessment strategy of “Do not reassess closed cases” then the Dependency Manager will remove any entries for cases that are closed as they will no longer be needed to support recalculations.

This document will outline a potential strategy to archive Dependency table entries for cases that are archived before they are closed.

3 Dependency table

When a client of the Dependency Manager identifies that one data item (a "dependent") depends on another (a "precedent"), then that client can request that the Dependency Manager stores the dependency as a row on this entity.

The Dependency manager uses the data on this table to identify which dependents are affected by changes in one or more precedents, which are stored as Precedent Change Items in a Precedent Change Set.

The values for 'dependentType' and 'dependentID' will vary depending on the client type. For example, if the client is the Eligibility and Entitlement Engine, then the 'dependentType' is 'Case Assessment Determination Result' and the 'dependentID' will be the 'caseID' for the case.

Attribute	Description
dependencyID	The unique identifier of the record .
dependentType	The type of the dependent.
dependentID	The identifier of the dependent (for the respective dependentType).
precedentType	The type of the precedent.
precedentID	The identifier of the precedent (for the respective precedent Type).

4 Identification of Table Rows for Archiving

To identify which rows in the Dependency table are associated with a given case, firstly check which 'dependentType's are configured on the system using the query below.

```
SELECT *

FROM CODETABLEITEM

WHERE tableName = 'DependentType';
```

To build the full list of Dependency record for a specific case, a separate query for each dependency type on the system needs to be run per case. The individual queries for the 'out of the box' dependent types are outlined in the following four sections .

Note a sample 'caseID' of '123' is used for illustrative purposes in these sample queries. Add the results of each of these queries together to get the full list of Dependency records for a specific case.

4.1 Case Assessment Determination Result

In this example the 'dependentType' is Case Assessment Determination Result and the 'dependentID' maps to the 'caseID' of a product delivery case that is being archived.

```
SELECT dependencyID
FROM Dependency
WHERE dependentType = 'CADETERRES'
AND dependentID = '123';
```

Add the list of 'dependencyID's resulting from this query to the full list of Dependency records associated with the specified case.

4.2 Advice Context

In this example the 'dependentType' is Advice Context and the 'dependentID' maps to the 'adviceContextID' for a piece of advice associated with the case that is being archived.

```
SELECT dependencyID
FROM Dependency
  INNER JOIN  AdviceContext ON
  Dependency.dependentID = AdviceContext.adviceContextID
  INNER JOIN AdviceXMLResult ON
  AdviceContext.adviceContextID = AdviceXMLResult.adviceContextID
  INNER JOIN AdviceCaseIssue ON
  AdviceCaseIssue.adviceXMLResultID = AdviceXMLResult.adviceXMLResultID
WHERE Dependency.dependentType = 'ADVICECTXT'
AND AdviceCaseIssue.caseID = '123';
```

Add the list of 'dependencyID's resulting from this query to the full list of Dependency records associated with the specified case.

4.3 Conditional Verification Result

In this example the 'dependentType' is Conditional Verification Result and the 'dependentID' maps to a 'conditionalVerResultID'. This value identifies a row on the ConditionalVerificationResult table which will also have an 'evidenceDescriptorID'. This in turn can be used to identify the associated product delivery case.

```
SELECT Dependency.dependencyID
FROM Dependency
  INNER JOIN  ConditionalVerificationResult ON
  Dependency.dependentID=ConditionalVerificationResult.conditionalVerResultID
  INNER JOIN  EvidenceDescriptor ON
  ConditionalVerificationResult.evidenceDescriptorID= EvidenceDescriptor.evidenceDescriptorID
WHERE Dependency.dependentType = 'CVRESULT'
  AND EvidenceDescriptor.caseID='123';
```

Add the list of 'dependencyID's resulting from this query to the full list of Dependency records associated with the specified case.

4.4 Stored Attribute Value

For entries in the Dependency table where the dependent type is Stored Attribute Value, the 'dependentID' maps to the key of a row in the CREOLERuleAttributeValue table. Only values associated with Rule Objects using a DatabaseDataStorage strategy are stored on this table. In out of the box Cúram, only Rate Rule Objects use this strategy, therefore entries to the Dependency table with a 'dependentType' of 'STORED_AV' should **not** be archived.