Merative™ SPM Platform

Guidelines for Archiving Precedent Change Sets

Cúram v6.0.2 EP13 and 6.0.4.0 and later

Author: Patrick O'Brien Owner:
Merative SPM Platform Group

Dated 15 September 2017

Document Control Information

Revision Status: 2.0

Document Author: Patrick O'Brien

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	[03/06/2013]	Patrick O'Brien	Initial Version
2	[15/09/17]	Billy Dennigan	Appended section, Reassessment Aggregation Tables

Contents

Introduction

The purpose of this document is to provide high level guidance, to assist customers on archiving Precedent Change Set records from the application. The document will describe briefly the tables associated with these records, the relationships between them and the order in which they must be archived.

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

This document also describes the Reassessment Aggregation tables that were added in v6.0.4.7 and v6.2.

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.

Precedent Change Sets Overview

(Note: For a full description of precedent change sets see the guide InsideCuramEiligibiltyAndEntitlementUsingCER).

When certain data gets changed during a transaction, information on these changes is recorded as a precedent change set, which can then be processed either in deferred or batch mode, depending on the type of data that was changed (e.g. publishing a rule set change will be processed in batch).

When processing is finished on a precedent change set, its status is generally set to COMPLETE, but in certain cases it can be set to DEFTOBATCH, meaning the work was deferred to the batch processing, generally as the application decided there was too much work to do in a single deferred process, without affecting other parts of the application.

Once processing is complete, the records are considered to be logically deleted, as they are no longer used in normal system operation. However, they can provide valuable information about the work previously done by the application, and so they are not physically deleted.

That said, there is no reason that older records need to be maintained in a database, and so they can be safely archived to reduce storage and increase performance of the related tables and indexes. We would, however, recommend keeping between 2 and 4 weeks worth of data in these tables where possible to aid in any future system debugging requirements.

Precedent Change Set tables

There are a total of six tables to be dealt with, two related to the precedent change sets themselves, and four related to the processing of them through Curam deferred processing. When archiving records from these tables, it should be noted that there are foreign key relationships between them, so the tables should be dealt with in the following order:

Table Name	**Key Attribute**	**Description**
DPTicket	ticketID	A request to execute a deferred process.
DPProcessInstance	instanceID	An execution of a deferred process.
DPErrorInformation	instDataID	An error which occurred during execution of a deferred process.
WMInstanceData	wm_InstDataID	Execution data for a deferred process.
PrecedentChangeItem	precedentChangeItemID	An individual item within the precedent change set.

PrecedentChangeSet	precedentChangeSetID	A precedent change set in the application.

Identification of Table Rows for Archiving

Precedent change sets can be in a number of different states in the database, and the instructions for identifying the database rows to archive are slightly different for each. As such this section will deal with each status in turn.

The provided SQL is merely a sample to indicate how to identify the specific rows that should be archived, and should be adjusted as appropriate in terms of dates.

Note: The deferred processing tables DPTicket, DPProcessInstance, DPErrorInformation and WMInstanceData will also contain information about other deferred processes that have run in the system, and so care should be taken to only archive those records related to the precedent change sets being archived.

Completed Change Sets

These represent change sets where the work has been completed successfully, either via batch or deferred processing. Completed change sets will have no associated error information and so the DPErrorInformation table is not required to be archived for these records.

The following SQL queries will return the IDs of records in the specific table related to the precedent change sets processed and completed on or before the given date:

DPTicket

```
SELECT DPTicket.TicketID
FROM DPTicket
  INNER JOIN DPProcessInstance
  ON DPTicket.ProcessInstID=DPProcessInstance.InstanceID
  INNER JOIN WMInstanceData
  ON WMInstanceData.WM_InstDataID=DPProcessInstance.InstDataID
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE PrecedentChangeSet.CompletedDateTime <=
     to_date ('2013/03/09', 'yyyy/mm/dd')
```

```
  AND PrecedentChangeSet.Status = 'COMPLETE'
  AND
DPProcessInstance.ProcessName='PERFORM_DEFERRED_RECALCULATIONS_FROM_
PRECEDENT_CHANGE_SET';
```

DPProcessInstance

```
SELECT DPProcessInstance.InstanceID
FROM DPProcessInstance
  INNER JOIN WMInstanceData
  ON WMInstanceData.WM_InstDataID=DPProcessInstance.InstDataID
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE PrecedentChangeSet.CompletedDateTime <=
     to_date ('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'COMPLETE'
  AND
DPProcessInstance.ProcessName='PERFORM_DEFERRED_RECALCULATIONS_FROM_
PRECEDENT_CHANGE_SET';
```

WMInstanceData

```
SELECT WMInstanceData.WM_InstDataID
FROM WMInstanceData
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE PrecedentChangeSet.CompletedDateTime <=
     to_date ('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'COMPLETE';
```

PrecedentChangeItem

```
SELECT PrecedentChangeItem.PrecedentChangeItemID AS PrecedentChangeItemID
FROM PrecedentChangeSet
  INNER JOIN PrecedentChangeItem
  ON PrecedentChangeItem.PrecedentChangeSetID =
```

```
    PrecedentChangeSet.PrecedentChangeSetID
WHERE PrecedentChangeSet.CompletedDateTime <=
     to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'COMPLETE';
```

PrecedentChangeSet

```
SELECT PrecedentChangeSetID
FROM PrecedentChangeSet
WHERE CompletedDateTime <=
     to_date ('2013/03/09', 'yyyy/mm/dd')
  AND Status = 'COMPLETE';
```

Deferred to Batch Change Sets

In this situation, precedent change sets were being processed and the system decided to defer the work to the batch precedent change set. In doing so, it copied the work from the change set being processed to the open batch change set, and marked the current change set as 'Deferred to Batch'. As such, this change set is considered to be complete and so can be archived.

The SQL here is slightly different and so needs to be run and processed separately.

The following SQL queries will return the IDs of records in the specific table related to the precedent change sets processed and deferred to batch on or before the given date:

Note: The DEFTOBATCH status only exists in Cúram version 6.0.4.0 iFix8 and above.

DPTicket

```
SELECT DPTicket.TicketID
FROM DPTicket
  INNER JOIN DPProcessInstance
  ON DPTicket.ProcessInstID=DPProcessInstance.InstanceID
  INNER JOIN WMInstanceData
  ON WMInstanceData.WM_InstDataID=DPProcessInstance.InstDataID
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
```

```
    to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'DEFTOBATCH'
  AND
DPProcessInstance.ProcessName='PERFORM_DEFERRED_RECALCULATIONS_FROM_
PRECEDENT_CHANGE_SET';
```

DPProcessInstance

```
SELECT DPProcessInstance.InstanceID
FROM DPProcessInstance
  INNER JOIN WMInstanceData
  ON WMInstanceData.WM_InstDataID=DPProcessInstance.InstDataID
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
    to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'DEFTOBATCH'
  AND
DPProcessInstance.ProcessName='PERFORM_DEFERRED_RECALCULATIONS_FROM_
PRECEDENT_CHANGE_SET';
```

DPErrorInformation

```
SELECT InstDataID
FROM DPErrorInformation
  INNER JOIN WMInstanceData
  ON DPErrorInformation.InstDataID=WMInstanceData.WM_InstDataID
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
    to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'DEFTOBATCH';
```

WMInstanceData

```
SELECT WMInstanceData.WM_InstDataID
```

```
FROM WMInstanceData
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
      to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'DEFTOBATCH';
```

PrecedentChangeItem

```
SELECT PrecedentChangeItem.PrecedentChangeItemID AS PrecedentChangeItemID
FROM PrecedentChangeSet
  INNER JOIN PrecedentChangeItem
  ON PrecedentChangeItem.PrecedentChangeSetID =
     PrecedentChangeSet.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
      to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'DEFTOBATCH';
```

PrecedentChangeSet

```
SELECT PrecedentChangeSetID
FROM PrecedentChangeSet
WHERE SubmittedDateTime <= to_date ('2013/03/09', 'yyyy/mm/dd')
  AND Status = 'DEFTOBATCH';
```

Submitted Change Sets

Extreme care should be taken with regard to change sets in the status of 'SUBMITTED'. In general these are change sets that are currently being processed by either the batch or deferred processing and so should not be archived.

However the large number of them in your system indicates that many of these were from before the fix for Deferred to Batch processing and so represent change sets where the deferred processing failed. However, the work in these change sets would have been processed as part of a bulk reassessment run done periodically so they can be safely archived.

As these change sets can represent either current work or error conditions in the system, it is imperative that they be kept in the system for longer than the completed change sets, or deferred to batch change sets.

The following SQL queries will return the IDs of records in the specific table related to the precedent change sets that were submitted on or before the given date and have stayed in that state:

DPTicket

```
SELECT DPTicket.TicketID
FROM DPTicket
  INNER JOIN DPProcessInstance
  ON DPTicket.ProcessInstID=DPProcessInstance.InstanceID
  INNER JOIN WMInstanceData
  ON WMInstanceData.WM_InstDataID=DPProcessInstance.InstDataID
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
      to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'SUBMITTED'
  AND
DPProcessInstance.ProcessName='PERFORM_DEFERRED_RECALCULATIONS_FROM_
PRECEDENT_CHANGE_SET';
```

DPProcessInstance

```
SELECT DPProcessInstance.InstanceID
FROM DPProcessInstance
  INNER JOIN WMInstanceData
  ON WMInstanceData.WM_InstDataID=DPProcessInstance.InstDataID
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
      to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'SUBMITTED'
```

```
  AND
DPProcessInstance.ProcessName='PERFORM_DEFERRED_RECALCULATIONS_FROM_
PRECEDENT_CHANGE_SET';
```

DPErrorInformation

```
SELECT InstDataID
FROM DPErrorInformation
  INNER JOIN WMInstanceData
  ON DPErrorInformation.InstDataID=WMInstanceData.WM_InstDataID
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
      to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'SUBMITTED';
```

WMInstanceData

```
SELECT WMInstanceData.WM_InstDataID
FROM WMInstanceData
  INNER JOIN PrecedentChangeSet
  ON
PrecedentChangeSet.PrecedentChangeSetID=WMInstanceData.PrecedentChangeSetID
WHERE  PrecedentChangeSet.SubmittedDateTime <=
      to_date('2013/03/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'SUBMITTED';
```

PrecedentChangeItem

```
SELECT PrecedentChangeItem.PrecedentChangeItemID
FROM PrecedentChangeSet
  INNER JOIN PrecedentChangeItem
  ON PrecedentChangeItem.PrecedentChangeSetID =
    PrecedentChangeSet.PrecedentChangeSetID
WHERE PrecedentChangeSet.SubmittedDateTime <= to_date('2013/02/09', 'yyyy/mm/dd')
  AND PrecedentChangeSet.Status = 'SUBMITTED';
```

PrecedentChangeSet

```
SELECT PrecedentChangeSetID
FROM PrecedentChangeSet
WHERE SubmittedDateTime <=
    to_date ('2013/03/09', 'yyyy/mm/dd')
 AND Status = 'SUBMITTED';
```

Open Change Sets

Change sets with a status of OPEN should under no circumstances be removed from the system, as these are current change sets, and archiving them could cause serious issues in the system.

Reassessment Aggregation Tables

In versions 6.2 and 6.0.4.7, the introduction of the Reassessment Aggregation added

three database tables to the application: DependencyTrace, ReassessmentQueue and ReassessmentQueueControl.

For the DependencyTrace table, it is advised that you retain the most recent 'REASSESS' and 'BATCHFAIL' records for each case, because the records are used to display the last reassessed time for product delivery cases on the determinations screen.

The following SQL code selects the unique ID of records that can be considered for archiving:

```
SELECT dependencyTraceID FROM DependencyTrace WHERE dependencyTraceID NOT IN (
   -- filter the latest REASSESS and BATCHFAIL records for each caseID
   -- from the DependencyTrace table.
 SELECT dependencyTraceID
 FROM dependencyTrace
 INNER JOIN (
 SELECT  MAX(timestamp1) ts, caseID cid
 FROM dependencyTrace
 GROUP BY caseID
```

```
 ) ON
 timeStamp1 = ts AND caseID = cid
 WHERE (traceType = 'REASSESS')
     OR (traceType = 'BATCHFAIL')
 )
```

For the ReassessmentQueueControl table, it is advisable to retain the following records for each case:

- The record with the most recent lockTs value
- The record with the most recent completionTS value
- The record with the most recent lockTs value and a null completionTS value

The records are used to display the determination status of product delivery cases, for example, whether the cases are queued for reassessment, currently undergoing reassessment, and so on.

A soft foreign key connects the ReassessmentQueue table and the ReassessmentQueueControl table. The following code shows the definition of the soft foreign key:

```
ALTER TABLE ReassessmentQueue FOREIGN KEY (queueControlID)
REFERENCES ReassessmentQueueControl(ReassessmentQueueControlID);
```

The ReassessmentQueue table is a child of the ReassessmentQueueControl table. Therefore, delete records from the ReassessmentQueue table before you delete records from the ReassessmentQueueControl table.

The following SQL code shows how to select the unique ID of records to be considered for archiving from the ReassessmentQueue table and the ReassessmentQueueControl table:

```
SELECT queueControlID FROM ReassessmentQueue  WHERE queueControlID NOT IN
( SELECT qi
FROM (
( -- 1. gets the latest lockTS for each caseID.
SELECT reassessmentQueueControlID qi,  caseID, statusCode, lockTS
  FROM ReassessmentQueueControl
  INNER JOIN  ( SELECT  MAX(lockTS) ts, caseID cid FROM ReassessmentQueueControl
GROUP BY caseID) ON  lockTS = ts AND caseID = cid
) -- 1 .end
UNION
SELECT * FROM
( -- 2. gets the latest completionTS for each caseID.
```

```
SELECT reassessmentQueueControlID qi,  caseID, statusCode, completionTS
  FROM ReassessmentQueueControl
  INNER JOIN  ( SELECT  MAX(completionTS) ts, caseID cid FROM
ReassessmentQueueControl GROUP BY caseID) ON  completionTS = ts AND caseID = cid
) -- 2. end
UNION
SELECT * FROM
( -- 3. gets the most recent entries which have no completionTS.
SELECT reassessmentQueueControlID qi,  caseID, statusCode, completionTS
  FROM ReassessmentQueueControl
  INNER JOIN  ( SELECT  MAX(lockTS) ts, caseID cid FROM ReassessmentQueueControl
WHERE completionTS IS NULL GROUP BY caseID ) ON  lockTS = ts AND caseID = cid
) -- 3. end
)
) -- not in...

SELECT reassessmentQueueControlID FROM ReassessmentQueueControl WHERE
reassessmentQueueControlID NOT IN
( SELECT qi
FROM (
( -- 1. gets the latest lockTS for each caseID.
SELECT reassessmentQueueControlID qi,  caseID, statusCode, lockTS
  FROM ReassessmentQueueControl
  INNER JOIN  ( SELECT  MAX(lockTS) ts, caseID cid FROM ReassessmentQueueControl
GROUP BY caseID) ON  lockTS = ts AND caseID = cid
) -- 1 .end
UNION
SELECT * FROM
( -- 2. gets the latest completionTS for each caseID.
SELECT reassessmentQueueControlID qi,  caseID, statusCode, completionTS
  FROM ReassessmentQueueControl
  INNER JOIN  ( SELECT  MAX(completionTS) ts, caseID cid FROM
ReassessmentQueueControl GROUP BY caseID) ON  completionTS = ts AND caseID = cid
) -- 2. end
UNION
SELECT * FROM
( -- 3. gets the most recent entries which have no completionTS.
SELECT reassessmentQueueControlID qi,  caseID, statusCode, completionTS
  FROM ReassessmentQueueControl
```

INNER JOIN (SELECT MAX(lockTS) ts, caseID cid FROM ReassessmentQueueControl WHERE completionTS IS NULL GROUP BY caseID) ON lockTS = ts AND caseID = cid

) -- 3. end

)

) -- not in...

End of document