IBM Cúram Social Program Management

Cúram Batch Performance Mechanisms

Version 6.0.5

IBM Cúram Social Program Management

Cúram Batch Performance Mechanisms

Version 6.0.5

Note
Before using this information and the product it supports, read the information in "Notices" on page 21

Revised: May 2013

This edition applies to IBM Cúram Social Program Management v6.0 5 and to all subsequent releases unless otherwise indicated in new editions.

Licensed Materials - Property of IBM.

Contents

Figures

Tables

Chapter 1. Introduction

1.1 Introduction

This document gives an overview of the application functionality which allows both caching of batch data and execution of multiple instances of a single batch process. These enhancements were designed to improve the efficiency and scalability of Cúram batch processing.

Note that this guide should be read in conjunction with the `Cúram Batch Processing Guide`, which provides a description of all other aspects of Cúram Batch Processing.

1.1.1 Intended Audience

This document is intended for people interested in batch process performance mechanisms in the application.

1.2 What Batch Performance Mechanisms does the application Provide?

Over and above good design and development paradigms, the application provides two primary mechanisms for improving the performance and scalability of Batch Processes:

- Batch Streaming
- Caching of Batch Process Data

1.2.1 Batch Streaming

Batch Streaming refers to the application support for the concurrent execution of multiple instances of batch processes. Each logical batch process in the application (for example, GenerateInstructionLineItems) is represented by two physical batch executables. The first, the 'Chunker', divides the record set to be processed into a number of subsets or 'chunks', based on a 'chunk size' parameter set via system properties. The second, the 'Stream', processes these chunks. The Stream processes each record in a chunk, commits the result, and then looks for another chunk to process. Multiple instances of the Stream can execute in parallel.

By utilizing this Batch Streaming mechanism, Cúram batch processes can employ all of the available processing power of their host machine(s). Ultimately, this allows for the processing of more records in a given time period than a single instance of a batch process would allow.

Chapter 2 discusses the architecture of this mechanism.

1.2.2 Caching of Batch Process Data

Cúram batch processes can also avail of transaction level data caching. Utilization of this mechanism can greatly reduce the volume of database I/O required for batch process execution. A good example of the performance savings that this provides is when an error is encountered when processing a record, requiring it to be skipped or excluded. Caching of data in such situations (where processing effectively needs to restart without including a particular record) greatly improves operational efficiency.

Chapter 3 provides a description of the caches available to Cúram batch process developers.

1.3 Cúram Batch Processes using Batch Streaming

The above streaming and caching mechanisms are used in the processing of the following batch programs:

- Determine Product Delivery Eligibility
- Generate Instruction Line Items
- Generate Instruments
- CREOLE Bulk Case Chunk Reassessment By Product
- Apply Product Reassessment Strategy
- Perform Batch Recalculations From Precedent Change Set

Chapter 4 details the structure of these batch programs with respect to Batch Streaming.

1.4 Operation of Streamed Batch Processes

Batch Streaming introduces a number of operational considerations, including:

- Command-line execution of streamed batch processes
- Properties introduced by Batch Streaming
- Exception processing in streamed batch processes

Chapter 5 discusses these considerations.

Chapter 2. Batch Streaming Architecture

2.1 Introduction

This chapter describes the architecture underlying batch streaming in the application.

A simple overview of the architecture is included in the figure below. The mechanism is based on the concept of segmenting data to be processed into subsets or 'chunks'. Once segmented, an arbitrary number of batch processes then can operate in parallel on these chunks, performing identical processing on each constituent record in each chunk. In this way, with the appropriate configuration of the number and distribution of the batch processes, the use of resources used can be maximized in the most efficient manner for each process.

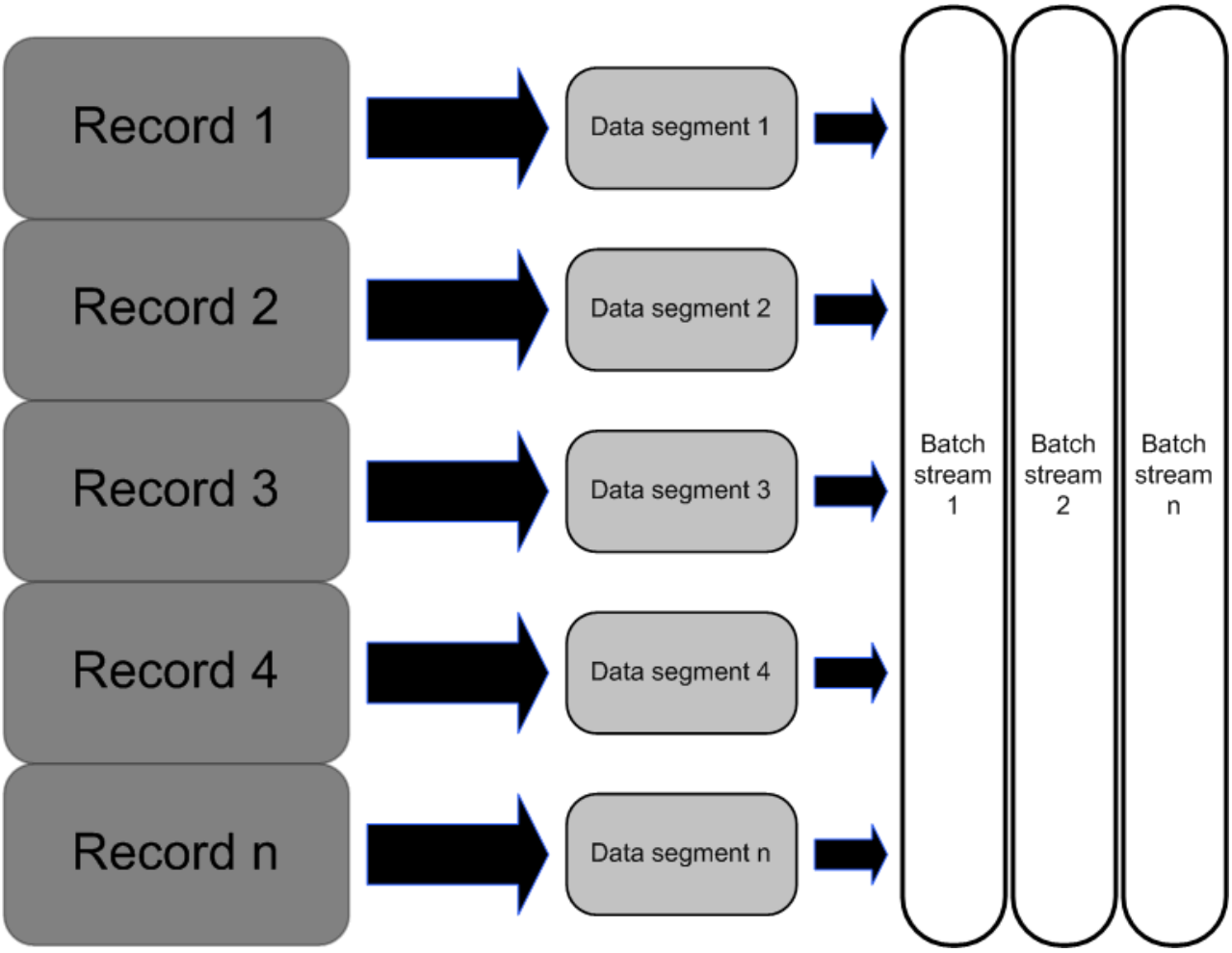

Figure 1. Streaming Architecture Overview

2.2 Architectural Details

As mentioned in the introduction, each logical Batch Process in composed of two physical batch executables: the 'Chunker' and the 'Stream'.

2.2.1 The Chunker

The 'Chunker' is the batch process executable which identifies the records to be processed. This process constructs 'chunks' of these records and writes them to the database, in effect assembling them in a queuing table called BatchProcessChunk. This table is populated at the beginning of batch processing, and each set of records to be processed is identified on this table by a chunk ID. Note that this assembly is transactional, and must succeed before any Streams can start their processing.

In addition to creating the chunks, the Chunker waits for all chunks to be processed by the Stream(s) and produces a summary report when they are all complete. In most cases only one instance of the Chunker is required for each batch process. Note that if the chunker fails after chunk assembly, it is possible to just restart it even if streaming has already commenced.

2.2.2 The Stream

The Stream is the batch process executable which performs the appropriate business functionality on each chunk. Each instance of a Stream operates on one chunk at a time, executing business processing on each record in the chunk in turn, and updating the chunk record with summary information once processing is complete. When complete, the records are marked as processed. If further chunks are available, processing starts again and the streams pick up another chunk.

Two important elements of this processing are as follows:

1. Each chunk is processed in a separate database transaction providing commit-point processing. This ensures that once a chunk is successfully processed, there will be no need to reprocess its constituent records if other chunks do not succeed for any reason.
2. Because processing of chunks is transactional, problem records can be excluded from the chunk during processing. For instance, if there is a lock contention during the processing of a chunk, one of the records may not be able to be processed at that point in time. In this instance, the work done by the chunk will be rolled back and the problem record removed. Processing of the chunk can then start again. Note that use of transaction level caching can greatly reduce the database I/O in this type of situation (see Chapter 3, “Data Caching,” on page 7 for more details).

   When all the chunks have been processed, a final search is done for any remaining unprocessed records. One final attempt is made to process these. These unprocessed chunks are processed serially. Streaming is not supported here to mitigate as far as possible against database contention and concurrency problems. This final search is optional, and is controlled by the 'chunkMainParameters' parameter of the runChunkMain method on the BatchStreamHelper in question.

   When this process is completed, a notification containing the details of those records processed and those not processed is sent. The recipient of this notification is defined by appropriate coding of the sendBatchReport method of the chunker batch process.

Note: No differentiation is made by the Batch Streaming environment between records remaining unprocessed because of technical issues, and those which were skipped for business reasons by the batch process. It will be left up to the batch administrator or user to examine all outputs to determine any cause of failures.

2.2.3 Additional Information

A more detailed diagram of the batch streaming architecture is included below. Two additional elements are of note here:

1. Chunk Key

   This table (called BatchChunkKey) is essentially used as a key server, allowing chunks to be "served" up to individual streams without creating contention on the chunk table itself. It is also worth noting that the value of the next key available can be examined to determine the progress of the batch program.
2. Batch params

The details of the parameters passed into the Chunker batch process are stored to make them available to each stream without being re-entered. This table, called BatchProcess, also contains the total number of chunks written, along with some other information about the Batch Process.

Note: The batch streaming architecture also supports the dynamic addition of streams while the batch process is being run, subject to the appropriate hardware being available to execute them.

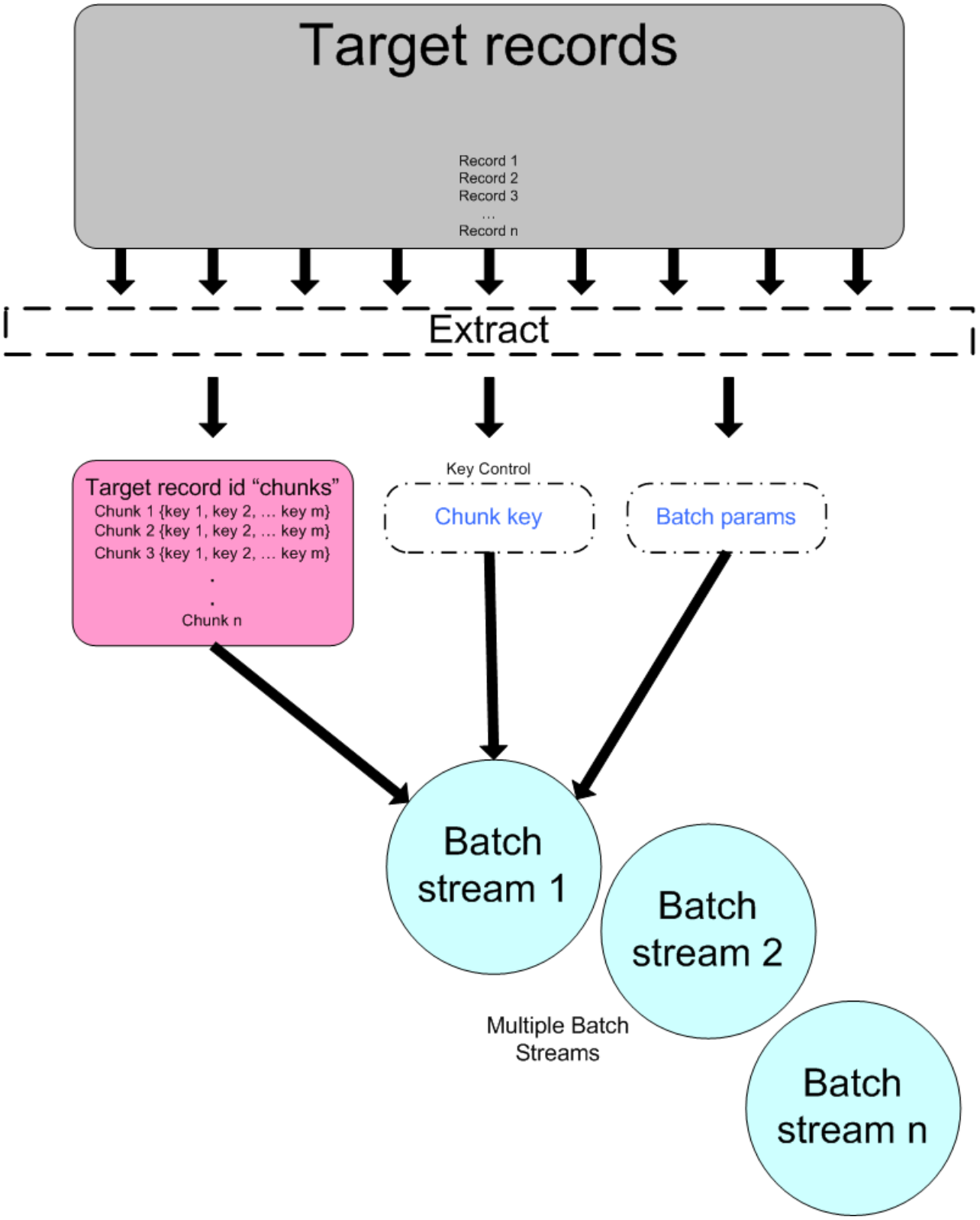

Figure 2. Streaming Architecture Details

Chapter 3. Data Caching

3.1 Introduction

The second batch performance mechanism provided by the application addresses the issue of database I/O contention. Improvements to database I/O in batch processing are always worth making, especially as batch windows reduce and the case and client loads increase. To this end a number of in-memory caches have been introduced for core Cúram entities which are available for re-use by Cúram batch processes.

Note that data which is accessed repeatedly during the eligibility processing is not limited to that stored in core Cúram entities. As a result, consideration should be given by customers to the caching of custom entities (e.g. evidence) which are accessed as part of this process.

It is also worth noting that these caches have been constructed so that they cannot be used in on-line mode. When on-line, because the application server is in control of the thread scheduling, the consistency between the cached data and that on the database cannot be guaranteed.

3.2 Core Cúram Entity Caches

Caches have been implemented for the following core Cúram Entities:

- CaseEvidenceTree

  This entity is one of the constituents of the Case Evidence Tree evidence maintenance solution. The caching of this entity has been incorporated into the CaseEvidenceAPI class and should not need to be accessed directly.
- CaseEvidenceGroupLink

  This entity is one of the constituents of the Case Evidence Tree evidence maintenance solution. The caching of this entity has been incorporated into the CaseEvidenceAPI class and should not need to be accessed directly.
- AttributedEvidence

  This entity is part of the Evidence maintenance solution. The caching of this entity has been incorporated into the Evidence Controller class and should not need to be accessed directly.
- CaseHeader

  This stand alone cache is implemented in the CachedCaseHeader class. Referencing this class rather than the CaseHeader entity directly will allow your processing take advantage of this cache.
- ConcernRole

  This stand alone cache is implemented in the CachedConcernRole class. Referencing this class rather than the ConcernRole entity directly will allow your processing take advantage of this cache.
- CaseNomineeProdDelPattern

  This stand alone cache is implemented in the CachedCaseNomineeProdDelPattern class. Referencing this class rather than the CaseNomineeProdDelPattern entity directly will allow your processing take advantage of this cache.
- CaseParticipantRole

  This stand alone cache is implemented in the CachedCaseParticipantRole class. Referencing this class rather than the CaseParticipantRole entity directly will allow your processing take advantage of this cache.
- CaseRelationship

This stand alone cache is implemented in the CachedCaseRelationship class. Referencing this class rather than the CaseRelationship entity directly will allow your processing take advantage of this cache.

- CaseStatus

  This stand alone cache is implemented in the CachedCaseStatus class. Referencing this class rather than the CaseStatus entity directly will allow your processing take advantage of this cache.

- ConcernRoleRelationship

  This stand alone cache is implemented in the CachedConcernRoleRelationship class. Referencing this class rather than the ConcernRoleRelationship entity directly will allow your processing take advantage of this cache.

- FinancialCalendar

  This stand alone cache is implemented in the CachedFinancialCalendar class. Referencing this class rather than the FinancialCalendar entity directly will allow your processing take advantage of this cache.

- Person

  This stand alone cache is implemented in the CachedPerson class. Referencing this class rather than the Person entity directly will allow your processing take advantage of this cache.

- Product

  This stand alone cache is implemented in the CachedProduct class. Referencing this class rather than the Product entity directly will allow your processing take advantage of this cache.

- ProductDelivery

  This stand alone cache is implemented in the CachedProductDelivery class. Referencing this class rather than the ProductDelivery entity directly will allow your processing take advantage of this cache.

- ProductDeliveryCertDiary

  This stand alone cache is implemented in the CachedProductDeliveryCertDiary class. Referencing this class rather than the ProductDeliveryCertDiary entity directly will allow your processing take advantage of this cache.

- ProductDeliveryPattern

  This stand alone cache is implemented in the CachedProductDeliveryPattern class. Referencing this class rather than the ProductDeliveryPattern entity directly will allow your processing take advantage of this cache.

- ProductDeliveryPatternInfo

  This stand alone cache is implemented in the CachedProductDeliveryPatternInfo class. Referencing this class rather than the ProductDeliveryPatternInfo entity directly will allow your processing take advantage of this cache.

- ProductRulesLink

  This stand alone cache is implemented in the CachedProductRulesLink class. Referencing this class rather than the ProductRulesLink entity directly will allow your processing take advantage of this cache.

- ProviderLocation

  This stand alone cache is implemented in the CachedProviderLocation class. Referencing this class rather than the ProviderLocation entity directly will allow your processing take advantage of this cache.

- RateTable

  This cache has been incorporated into the RateTable service layer class and should not need to be accessed directly.

- SupplierReturnHeader

This stand alone cache is implemented in the CachedSupplierReturnHeader class. Referencing this class rather than the SupplierReturnHeader entity directly will allow your processing take advantage of this cache.

Chapter 4. Cúram Batch Processes using Streaming

4.1 Introduction

This chapter describes the Core Cúram batch processes which implement streaming and caching, together with their operational characteristics. Each process has been provided as two executables, as described in the previous two chapters.

4.2 DetermineProductDeliveryEligibility

This batch process takes "Approved" cases and runs the determine eligibility process. Two batch executables are provided for this batch process:

- DetermineProductDeliveryEligibility

  This executable is the Chunker for this process. It identifies all cases which are “Approved” and writes their caseIDs to the chunks. This process also accepts a product identifier as an optional input parameter, which is used to limit the cases selected to those which are instances of a particular product.
- DetermineProductDeliveryEligibilityStream

  This program is the Stream for this process. It runs the determine eligibility process for each case and stores the results on the database.

4.3 GenerateInstructionLineItems

This batch program takes Financial Components due to be processed, reassesses the case for the period to be paid and generates the appropriate Instruction Line Item records. Two batch executables are provided for this program:

- GenerateInstructionLineItems

  This executable is the Chunker for this process. It identifies all cases with Financial Components due to be processed and writes their caseIDs to the chunks. This process also accepts a set of optional input parameters, a 'from' date, a 'to' date and a delivery method, which are used to limit the cases selected to be processed.
- GenerateInstructionLineItemsStream

  This program is the Stream for this process. It runs the determine eligibility process for the period to be paid for each case and generates all relevant Instruction Line Items.

4.4 GenerateInstruments

This batch program takes Instruction Line Items due to be processed and generates Payment Instrument records. Two batch executables are provided for this program:

- GenerateInstruments

  This executable is the Chunker for this process. It identifies all nominees with Instruction Line Items due to be processed and writes their nomineeIDs to the chunks.
- GenerateInstrumentsStream

  This program is the Stream for this process. It generates Payment Instruments for each nominee.

4.5 CREOLEBulkCaseChunkReassessmentByProduct

This batch process takes "Active" CER cases and runs the case reassessment process on them. Two batch executables are provided for this batch process:

Important: As this process will cause reassessment of all cases of the specified type, it may cause a lot of unneccessary reassessments. Where appropriate, a new batch process should be written in order to more precisely identify the cases that require reassessment, especially when the cases are spread across a range of products. For a full explanation of how to write an appropriate batch process see the Inside Cúram Eligibility and Entitlement Using Cúram Express Rules guide.

- CREOLEBulkCaseChunkReassessmentByProduct

  This executable is the Chunker for the bulk case reassessment process. It identifies all cases which are "Active" and writes their caseIDs to the chunks. The bulk case reassessment also accepts a product identifier as an optional input parameter, which is used to limit the cases selected to those which are instances of a particular product.
- CREOLEBulkCaseChunkReassessmentStream

  This program is the Stream for the bulk case reassessment process. It runs the case reassessment process for each case and, if the determination has changed, it stores the new determination and supersedes the previous one.

4.6 ApplyProductReassessmentStrategy

This batch process checks all the cases for a CER-based product whose reassessment strategy has changed.

- ApplyProductReassessmentStrategy

  This executable is the Chunker for the Apply Product Reassessment Strategy process. It takes a product ID as input, and for that product identifies all product delivery cases and writes their caseIDs to the chunks.
- ApplyProductReassessmentStrategyStream

  This program is the Stream for the Apply Product Reassessment Strategy process. For each product delivery case, the program checks to see if the case's support for reassessment has changed due to the change in the product's reassessment strategy.

For each product delivery case for the product:

- if the case was not reassessable under the old strategy but becomes reassessable under the new strategy, then an assessment is performed on the case to build up the dependency records for the case's determination result;
- if the case was reassessable under the old strategy but is no longer reassessable under the new strategy, then the dependency records for the determination result are removed;
- otherwise no action is performed on the case.

See the Inside Cúram Eligibility and Entitlement Using Cúram Express Rules guide for more detail.

4.7 PerformBatchRecalculationsFromPrecedentChangeSet

This batch process analyzes a set of changes to precedent data and recalculates all dependents potentially affected by any of those precedent changes.

- PerformBatchRecalculationsFromPrecedentChangeSet

  This executable is the Chunker for the Perform Batch Recalculations From Precedent Change Set process. It takes a dependent type as input, examines the most-recent precedent change set which has been submitted for batch processing, identifies the unique set of dependents (for the type input) potentially affected by any of the precedent change items in the submitted precedent change set, and writes their dependent IDs to the chunks.
- PerformBatchRecalculationsFromPrecedentChangeSetStream

  This program is the Stream for the Perform Batch Recalculations From Precedent Change Set. For each dependent identified, the dependent is recalculated in a manner appropriate to its type, e.g. if the dependent identifies a CER-based case, the case will be reassessed.

For full details on this batch process, see "The Dependency Manager" in the Cúram Express Rules Reference Guide.

Chapter 5. Operation of Streamed Batch Processes

5.1 Introduction

This chapter details various operational considerations which apply when deploying the streamed batch processes in the application. Note that similar considerations should also apply to customer-written streamed batch processes.

5.2 Running Batch Executables

To launch the batch executables on a machine the following command can be used:

```
ant -f app_batchlauncher.xml -Dbatch.username=superuser -Dbatch.program=<method name>
```

Where the method name is the appropriate one from the list below:

Table 1. Method Names for batch executables

Executable	Method Name
DetermineProductDeliveryEligibility	curam.core.intf.DetermineProductDeliveryEligibility.process
DetermineProductDeliveryEligibilityStream	curam.core.intf.DetermineProductDeliveryEligibilityStream.process
GenerateInstructionLineItems	curam.core.intf.GenerateInstructionLineItems.processAllFinancialCompone
GenerateInstructionLineItemsStream	curam.core.intf.GenerateInstructionLineItemsStream.process
GenerateInstruments	curam.core.intf.GenerateInstruments.processInstructionLineItemsDue
GenerateInstrumentsStream	curam.core.intf.GenerateInstrumentsStream.process
CREOLEBulkCaseChunkReassessmentByProduct	curam.core.sl.infrastructure.assessment.intf.CREOLEBulkCaseChunkReass
CREOLEBulkCaseChunkReassessmentStream	curam.core.sl.infrastructure.assessment.intf.CREOLEBulkCaseChunkReass
ApplyProductReassessmentStrategy	curam.core.sl.infrastructure.assessment.intf.ApplyProductReassessmentStr
ApplyProductReassessmentStrategyStream	curam.core.sl.infrastructure.assessment.intf.ApplyProductReassessmentStr
PerformBatchRecalculationsFromPrecedentChangeSet	curam.dependency.intf.PerformBatchRecalculationsFromPrecedentChange
PerformBatchRecalculationsFromPrecedentChangeSetStream	curam.dependency.intf.PerformBatchRecalculationsFromPrecedentChange

So for example to run the DetermineProductDeliveryEligibilityStream process the command would be:

```
ant -f app_batchlauncher.xml -Dbatch.username=superuser
-Dbatch.program=curam.core.intf.DetermineProductDeliveryEligibilityStream.process
```

Note that it is possible to use the BatchLauncher to run the batch executables; however the queued processes will be run sequentially.

5.3 Environment variables

The environment variables listed below control the operation of the various batch performance mechanisms described in the previous chapters. It is important to note that while the tuning of these parameters is key to achieving the best performance when running batch processes, it is also possible to compromise their performance by incorrect tuning of these parameters. It is therefore advised that the impact of changes to each parameter be assessed individually to ensure that it has the expected affect on performance.

5.3.1 General Batch streaming

The following environment variables control the generic batch streaming infrastructure behavior:

- curam.batch.streams.batchprocessreadwaitinterval

  The interval (in milliseconds) for which a batch stream will wait before retrying when reading the batch process table.
- curam.batch.streams.chunkkeyreadwaitinterval

  The interval (in milliseconds) for which a batch stream will wait before retrying when reading the chunk key table.
- curam.batch.streams.scanforunprocessedchunkswaitinterval

  The interval (in milliseconds) for which the main batch process (chunker) will wait before trying to scan for unprocessed chunks, once the value in the chunk key table has exceeded the number of chunks.

5.3.2 General Caching

The following environment variables control generic caching behavior:

- curam.batch.caching.buffersize

  Batch process caches using circular buffers will use this value to set the initial buffer size.

5.3.3 Determine Product Delivery Eligibility

The following environment variables control the behavior of the Determine Product Delivery Eligibility program:

- curam.batch.determineproductdeliveryeligibility.chunksize

  The number of cases in each chunk that will be processed by the Determine Product Delivery Eligibility batch program.
- curam.batch.determineproductdeliveryeligibility.dontrunstream

  Indicates whether the Determine Product Delivery Eligibility batch program should sleep while waiting for processing to be completed.
- curam.batch.determineproductdeliveryeligibility.chunkkeywaitinterval

  The interval (in milliseconds) for which the Determine Product Delivery Eligibility batch program will wait before retrying when reading the chunk key table.
- curam.batch.determineproductdeliveryeligibility.unprocessedchunkwaitinterval

  The interval (in milliseconds) for which the Determine Product Delivery Eligibility batch program will wait before retrying when reading the chunk table.
- curam.batch.determineproductdeliveryeligibility.processunprocessedchunk

  Indicates whether the Determine Product Delivery Eligibility program should attempt to process any unprocessed chunks found after all the streams have completed.

5.3.4 Generate Instruction Line Items

The following environment variables control the behavior of the Generate Instruction Line Items process:

- curam.batch.generateinstructionlineitems.chunksize

  The number of cases in each chunk that will be processed by the Generate Instruction Line Items batch process.
- curam.batch.generateinstructionlineitems.dontrunstream

  Indicates whether the Generate Instruction Line Items batch program should sleep while waiting for the processing to be completed.
- curam.batch.generateinstructionlineitems.chunkkeywaitinterval

  The interval (in milliseconds) for which the Generate Instruction Line Items batch process will wait before retrying when reading the chunk key table.

- curam.batch.generateinstructionlineitems.unprocessedchunkwaitinterval

  The interval (in milliseconds) for which the Generate Instruction Line Items batch process will wait before retrying when reading the chunk table.
- curam.batch.generateinstructionlineitems.processunprocessedchunk

  Indicates whether the Generate Instruction Line Items program should attempt to process any unprocessed chunks found after all the streams have completed.
- curam.batch.generateinstructionlineitems.dontreassesscase

  Indicates whether the Generate Instruction Line Items program should skip reassessment of the case prior to payment.

5.3.5 Generate Instruments

The following environment variables control the behavior of the Generate Instruments process:

- curam.batch.generateinstruments.chunksize

  The number of cases in each chunk that will be processed by the Generate Instruments batch process.
- curam.batch.generateinstruments.dontrunstream

  Indicates whether the Generate Instruments batch process should sleep while waiting for the processing to be completed.
- curam.batch.generateinstruments.chunkkeywaitinterval

  The interval (in milliseconds) for which the Generate Instruments batch process will wait before retrying when reading the chunk key table.
- curam.batch.generateinstruments.unprocessedchunkwaitinterval

  The interval (in milliseconds) for which the Generate Instruments batch process will wait before retrying when reading the chunk table.
- curam.batch.generateinstruments.processunprocessedchunk

  Indicates whether the Generate Instruments program should attempt to process any unprocessed chunks found after all the streams have completed.

5.3.6 CREOLE Bulk Case Chunk Reassessment By Product

The following environment variables control the behavior of the CREOLE Bulk Case Chunk Reassessment By Product program, and its associated Stream process (CREOLE Bulk Case Chunk Reassessment Stream):

- curam.batch.creolebulkcasechunkreassessment.chunksize

  The number of cases in each chunk that will be processed by the CREOLE Bulk Case Chunk Reassessment Stream program.
- curam.batch.creolebulkcasechunkreassessment.dontrunstream

  Indicates whether the CREOLE Bulk Case Chunk Reassessment By Product batch program should sleep while waiting for processing to be completed.
- curam.batch.creolebulkcasechunkreassessment.chunkkeywaitinterval

  The interval (in milliseconds) for which the CREOLE Bulk Case Chunk Reassessment By Product batch program will wait before retrying when reading the chunk key table.
- curam.batch.creolebulkcasechunkreassessment.unprocessedchunkwaitinterval

  The interval (in milliseconds) for which the CREOLE Bulk Case Chunk Reassessment By Product batch program will wait before retrying when reading the chunk table for unprocessed chunks.
- curam.batch.creolebulkcasechunkreassessment.processunprocessedchunk

  Indicates whether the CREOLE Bulk Case Chunk Reassessment By Product program should attempt to process any unprocessed chunks found after all the streams have completed.

5.3.7 Apply Product Reassessment Strategy

The following environment variables control the behavior of the Apply Product Reassessment Strategy program, and its associated Stream process (Apply Product Reassessment Strategy Stream):

- curam.batch.applyproductreassessmentstrategy.chunksize

  The number of cases in each chunk that will be processed by the Apply Product Reassessment Strategy batch program.
- curam.batch.applyproductreassessmentstrategy.dontrunstream

  Indicates whether the Apply Product Reassessment Strategy batch program should sleep while waiting for the processing to be completed (rather than run a stream in its context)
- curam.batch.applyproductreassessmentstrategy.chunkkeywaitinterval

  The interval (in milliseconds) for which the Apply Product Reassessment Strategy batch program will wait before retrying when reading the chunk key table.
- curam.batch.applyproductreassessmentstrategy.unprocessedchunkwaitinterval

  The interval (in milliseconds) for which the Apply Product Reassessment Strategy batch program will wait before retrying when reading the chunk table for unprocessed chunks.
- curam.batch.applyproductreassessmentstrategy.processunprocessedchunk

  Indicates whether the Apply Product Reassessment Strategy program should process any unprocessed chunks found after all the streams have completed.

5.3.8 Perform Batch Recalculations From Precedent Change Set

The following environment variables control the behavior of the Perform Batch Recalculations From Precedent Change Set program, and its associated Stream process (Perform Batch Recalculations From Precedent Change Set Stream):

- curam.batch.performbatchrecalculationsfromprecedentchangeset.chunksize

  The number of dependents in each chunk that will be processed by the Perform Batch Recalculations From Precedent Change Set batch program.
- curam.batch.performbatchrecalculationsfromprecedentchangeset.dontrunstream

  Indicates whether the Perform Batch Recalculations From Precedent Change Set batch program should sleep while waiting for the processing to be completed (rather than run a stream in its context)
- curam.batch.performbatchrecalculationsfromprecedentchangeset.chunkkeywaitinterval

  The interval (in milliseconds) for which the Perform Batch Recalculations From Precedent Change Set batch program will wait before retrying when reading the chunk key table.
- curam.batch.performbatchrecalculationsfromprecedentchangeset.unprocessedchunkwaitinterval

  The interval (in milliseconds) for which the Perform Batch Recalculations From Precedent Change Set batch program will wait before retrying when reading the chunk table for unprocessed chunks.
- curam.batch.performbatchrecalculationsfromprecedentchangeset.processunprocessedchunk

  Indicates whether the Perform Batch Recalculations From Precedent Change Set program should process any unprocessed chunks found after all the streams have completed.

5.4 Error handling

Two key types of errors can occur when running the streamed batch programs:

- Skipped chunks

  These are reported when the batch process completes, together with an estimate of the total number of records which may have been affected. Re-running the batch process should process these chunks correctly, unless some fatal error is occurring during the batch processing. Note that skipped chunks are a relatively rare phenomenon; skipped records are far more likely.
- Skipped records

These are also reported when the batch program completes, and entries are added to the log files for the stream(s) which encountered the errors, detailing the error that occurred and the stack trace. There are two possible scenarios for this:

1. Some business error was encountered processing the record

   The status of the record will have been changed to remove it from the set of records to be processed and a task will have been created for the business owner. This takes the form of suspending the case and sending a task to the case owner.

2. Some technical error was encountered processing the record

   The status of the record will not have been changed by this event. The log file(s) should be examined to determine the problem, and the batch process re-run to process these records, once the issue has been resolved.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law.

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources.

IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing

application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/us/en/copytrade.shtml.

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

Printed in USA