Enrollment System (ES)

ES Batch Processes

Technical Specification

**

Department of Veterans Affairs

Office of Information and Technology (OIT)

Product Development

Documentation Version 1.2

Revision History

| Date | Revision | Description | Author |
| --- | --- | --- | --- |
| 01/2019 | 1.2 | Add VCE Recalculation Process | Dennis Doussan |
| 06/2018 | 1.1 | Add Activate System Parameters | Dennis Doussan |
| 06/2018 | 1.0 | Initial document | Trinadh Kavuri |

Table of Contents

[1 Introduction 1](#_Toc2248329)

[1.1 Purpose 1](#_Toc2248330)

[1.2 Scope 1](#_Toc2248331)

[2 System Components and Processes 2](#_Toc2248332)

[2.1 CloseApplicationHistoryProcess 2](#_Toc2248333)

[2.2 Pending Verification Mailing Confirmation File for ScheduledJob.PVCCMailingResponseConfirmationFileProcess 4](#_Toc2248334)

[2.3 Pending Verification Mailing Code 1 Reject File for PVMailingResponseCode1RejectFileProcess 7](#_Toc2248335)

[2.4 Pending Verification Mailing USPS Returned File for PVMailingResponseReturnedByUSPSFileProcess 10](#_Toc2248336)

[2.5 Pending Verification Mailing Error File for ScheduledJob.PVMailingResponseErrorFileProcess 12](#_Toc2248337)

[2.6 Activate VCE System Parameters 14](#_Toc2248338)

[2.7 VCE Parameters Changed Recalculation Batch Process 15](#_Toc2248339)

# Introduction

## Purpose

The purpose of this document is to document system components and processes for the Enrollment System (ES) batch processes.

## Scope

Technical specification describes system components and processes that change/add/edit the existing batch process or to create a new Batch job. Each new batch job added should discuss the transaction boundaries, error/exception handling, multi-threading and load handling.

# System Components and Processes

## CloseApplicationHistoryProcess

**Description:** Batch process will set enrollment applications to "Closed Application" status and apply an enrollment end date based on a file provided by the business which contains Integration Control Numbers (ICNs) and enrollment end dates for each application to be closed.

**Main Class:** CloseApplicationHistoryProcess

**Extends:** AbstractDataFileIncrementalProcess (Throttles reading to 25 records each iteration)

**Transaction Boundaries.**

1. A new transaction will be started for each record by delegating the work to

EligibilityEnrollmentServiceImpl. processCloseEnrollmentHistoryApplication(String vpid\_value, boolean isTriggerCloseLetter, Date enrollmentEndDate).

1. Process statistics will have their own transaction already configured as part of the framework.

**Exception Handling:** Any exception thrown out by the service will roll back the transaction, increment the error count and write the exception to the exception file. Exception file will have the same name as input file with a .exception extension.

**Input:**

1. **File naming pattern: CA\_\*.txt**
2. **Record Format: VPID^EndDate**
3. **Sample Record:** 0000001008795251V986965000000^20180607

**Close Application Batch Process List View.**

Figure 1: Close Application Batch Process List View

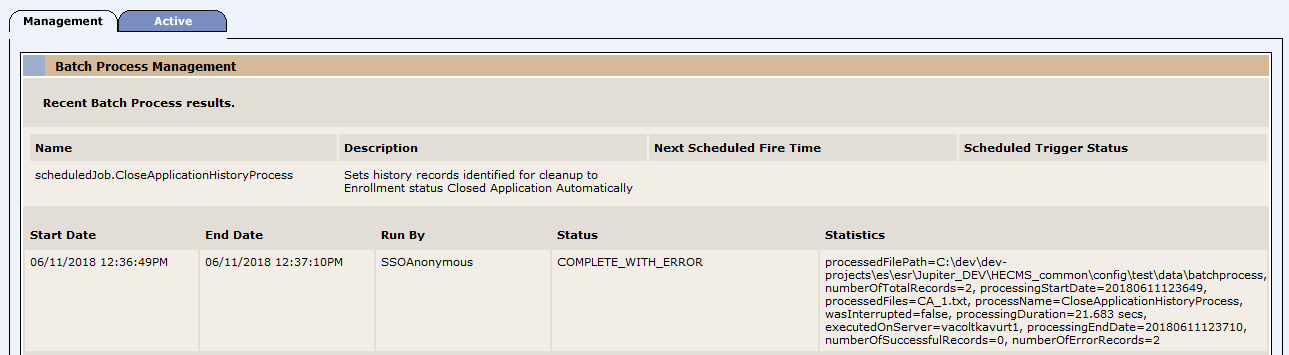
****

Figure 2: Close Application Batch Process History View

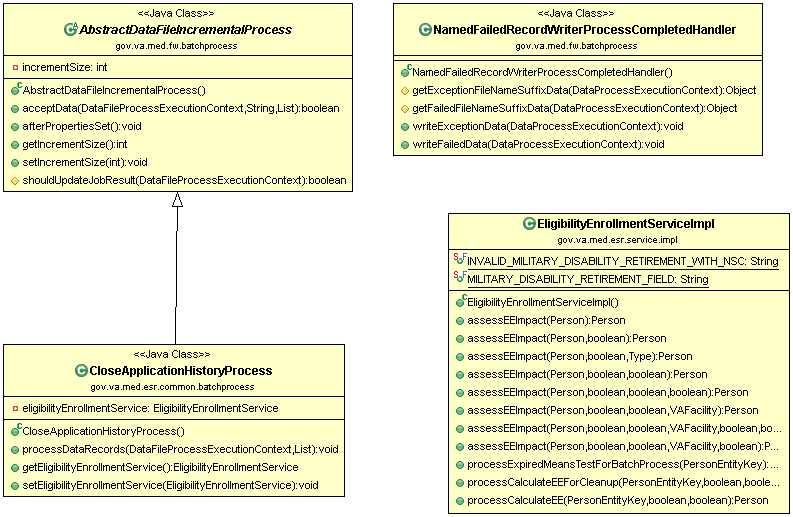


Figure 3: Class Diagram Close Application Batch Process

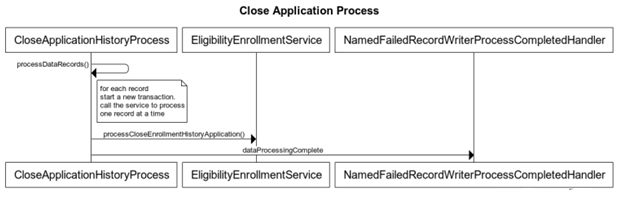


Figure 4: Sequence Diagram Close Application Batch Process

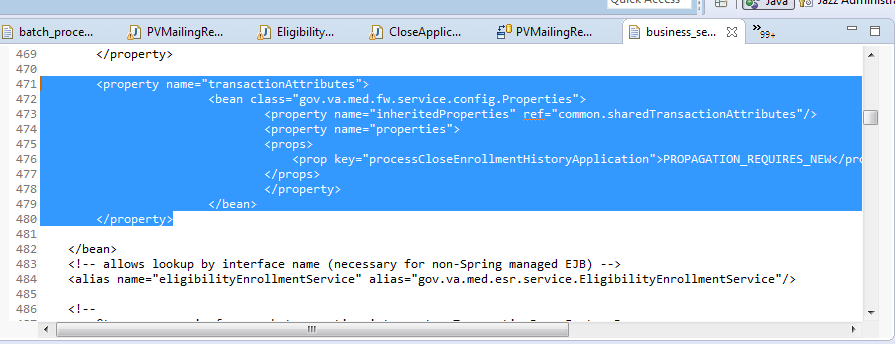


Figure 5: Spring configuration for transaction boundaries

## Pending Verification Mailing Confirmation File for ScheduledJob.PVCCMailingResponseConfirmationFileProcess

Description: Batch process will import successful mailing responses for the initial pending letters (742-650/742-651) and the new pre-closure letter (742-655) and display in the ES Previously Mailed and Available for Mailing tabs.

**Main Class:** PVMailingResponseConfirmationFileProcess

**Extends:** AbstractDataFileSingleRowIncrementProcess (Throttles reading to 25 records each iteration)

**Transaction Boundaries.**

1. A new transaction will be started for each record by delegating the work to

commsLogService.processConfDataRecord(dataRecord, daysToTriggerClock, daysToTriggerNotificationClock, fileName);

1. Process statistics will have their own transaction which are already configured as part of the framework.

**Exception Handling:** Any exception thrown out by the service will roll back the transaction, increment the error count and write the exception to the exception file. Exception file will have the same name as input file with a .exception extension.

**Multi-Threading:** This process run with seven threads with a task threshold of 10 tasks

**Input:**

1. **File naming pattern: PLC\_CMS\_confirmation\_file\_\*.txt**
2. **Record Format:** formNum^uniqueId^vpid^fullName^addressline1^addressline2^city^state^zip^datemailed
3. **Sample Record:**

Third column unique id is for processing on going letters.

742-652A^3699634862^1009146171V722148^LAURIE L FORBES^529 MAIN ST^^NEWPORT^VT^05855-4981^08252017

Unique id is missing to process historical pre-closure letters.

742-651^^1008795251V986965^LAURIE L FORBES^529 MAIN ST^^NEWPORT^VT^05855-4981^08252016

**Confirmation File Process Batch Process List View**

Figure 6: Confirmation File Process Batch Process List View

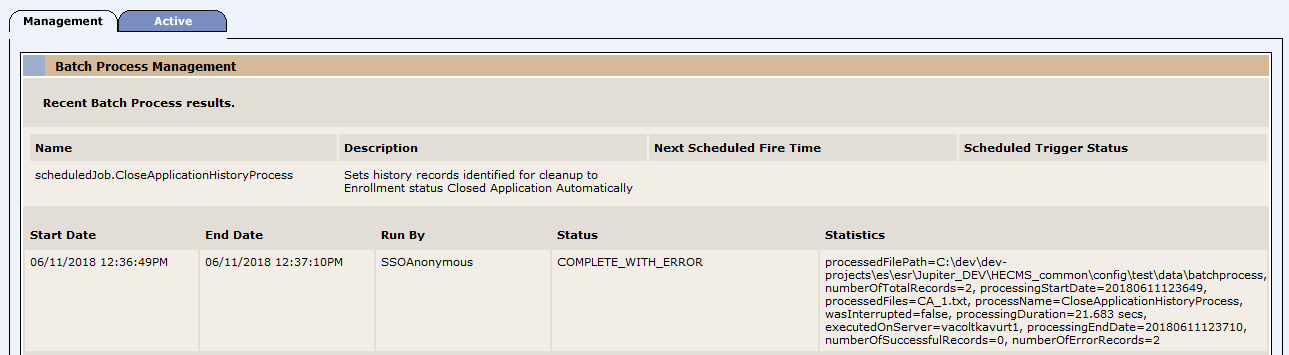
****

Figure 7: Confirmation File Process Batch Process History View

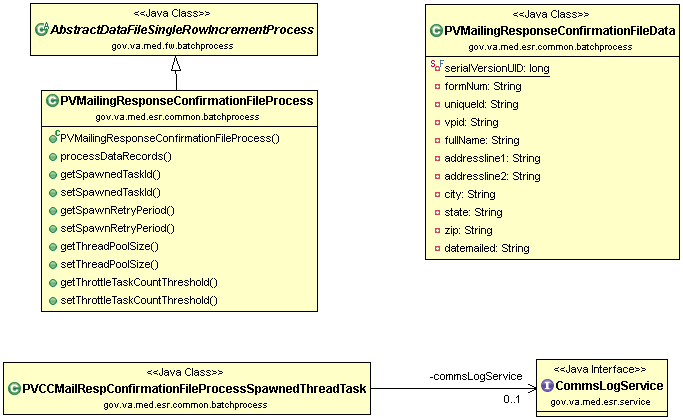


Figure 8: Class Diagram Confirmation File Batch Process

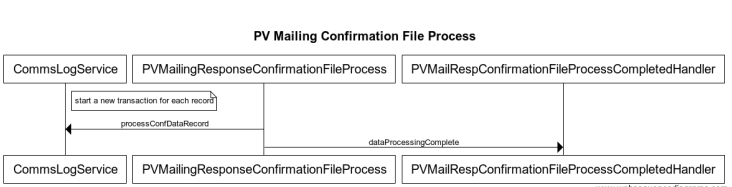


Figure 9: Sequence Diagram Confirmation File Batch Process

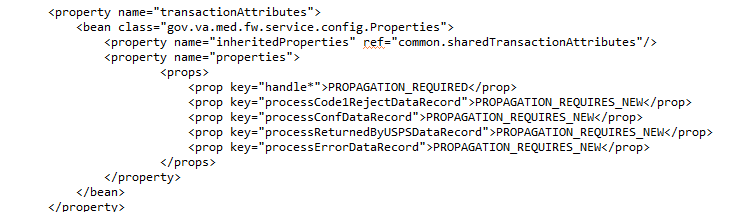


Figure 10: Spring configuration for transaction boundaries

## Pending Verification Mailing Code 1 Reject File for PVMailingResponseCode1RejectFileProcess

Description: Batch process will import Code1 Reject mailing responses for the initial pending letters (742-650/742-651) and the new pre-closure letter (742-655).

**Main Class:** PVMailingResponseCode1RejectFileProcess

**Extends:** AbstractDataFileSingleRowIncrementProcess (Throttles reading to 25 records each iteration)

**Transaction Boundaries.**

1. A new transaction will be started for each record by delegating the work to

commsLogService. processCode1RejectDataRecord(dataRecord, fileName);

1. Process statistics will have their own transaction which are already configured as part of the framework.

**Exception Handling:** Any exception thrown out by the service will roll back the transaction, increment the error count and write the exception to the exception file. Exception file will have the same name as input file with a .exception extension.

**Multi-Threading:** This process run with seven threads with a task threshold of 10 tasks.

**Input:**

1. **File naming pattern: PLC\_CMS\_USPS\_returns\_\*.txt**
2. **Record Format:** formNum^uniqueId^vpid^fullName^addressline1^addressline2^city^state^zip^datemailed^datereturned
3. **Sample Record:**

2nd column unique id is for processing on going letters.

742-652^3699623157^1017277454V797544^TESTEREAT CRMTESTEAT^9371 SIGNAL STATION DR^^MANASSAS^VA^20111-8258^08222017^08252017

Unique id is missing to process historical pre-closure letters.

742-652A^40125502958^1008782311V208987^LETTERTHREE TESTERTOO^123 TEST TRET^^STERLING^VA^20167^0822017^08252017

USPS Returned File Process Batch Process List View

Figure 11: USPS Returned File Process Batch Process List View

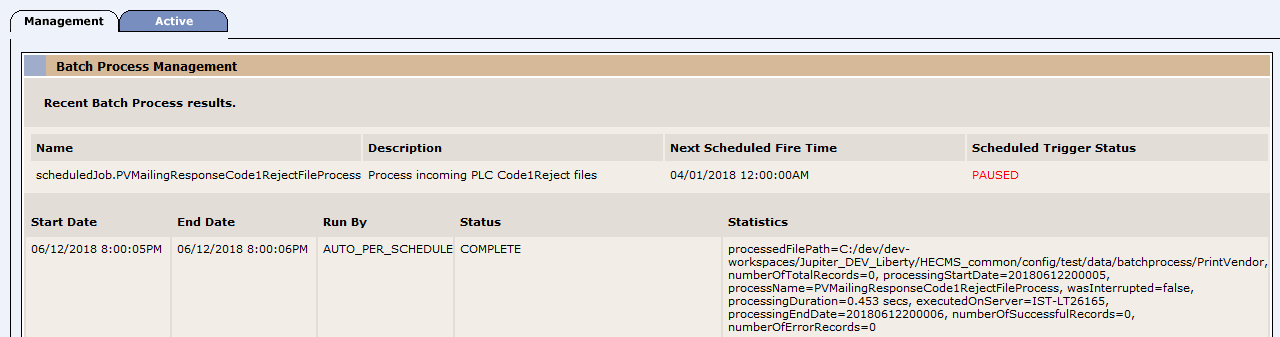


Figure 12: USPS Returned File Process Batch Process History View

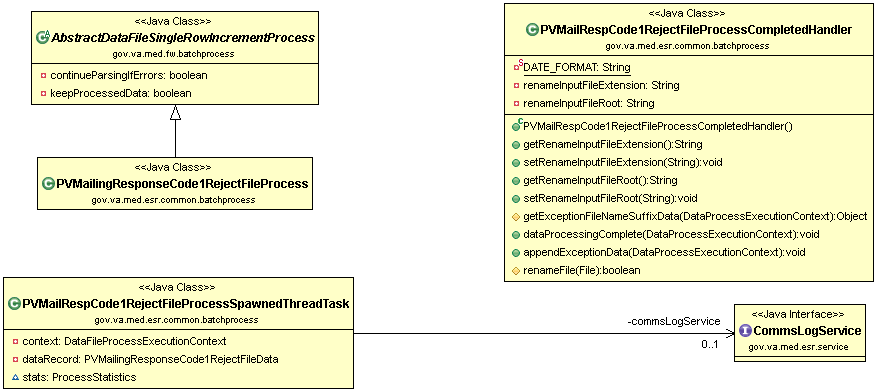


Figure 13: Class Diagram Code1 Reject File Batch Process

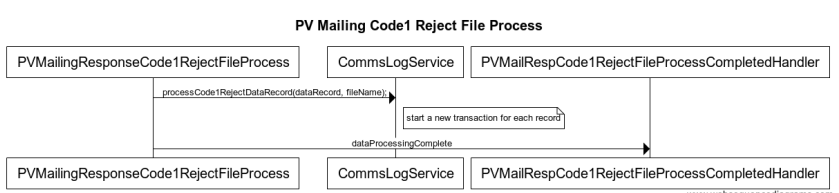


Figure 14: Sequence Diagram Code1 Reject File Batch Process

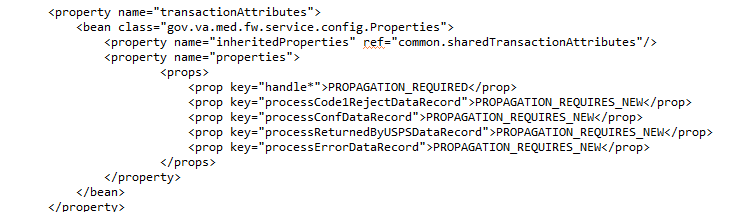


Figure 15: Spring configuration for transaction boundaries

## Pending Verification Mailing USPS Returned File for PVMailingResponseReturnedByUSPSFileProcess

Description: Batch Process will import USPS returned mailing responses for the initial pending letters (742-650/742-651) and the new pre-closure letter (742-655).

**Main Class:** PVMailingResponseReturnedByUSPSFileProcess

**Extends:** AbstractDataFileSingleRowIncrementProcess (Throttles reading to 25 records each iteration)

**Transaction Boundaries.**

1. A new transaction will be started for each record by delegating the work to

commsLogService. processReturnedByUSPSDataRecord(fileName, dataRecord);.

1. Process statistics will have their own transaction which are already configured as part of the framework.

**Exception Handling:** Any exception thrown out by the service will roll back the transaction, increment the error count and write the exception to the exception file. Exception file will have the same name as input file with a .exception extension.

**Multi-Threading:** This process run with seven threads with a task threshold of 10 tasks

**Input:**

1. **File naming pattern: PLC\_CMS\_USPS\_returns\_ \*.txt**
2. **Record Format:** formNum^uniqueId^vpid^fullName^addressline1^addressline2^city^state^zip^datemailed
3. **Sample Record:**

2nd column unique id is for processing on going letters.

742-652A^385984210^1013006426V772707^B

Unique id is missing to process historical pre-closure letters.

742-651^^1008795251V986965^B

USPS Returned File Process Batch Process List View

Figure 16: USPS Returned File Process Batch Process List View

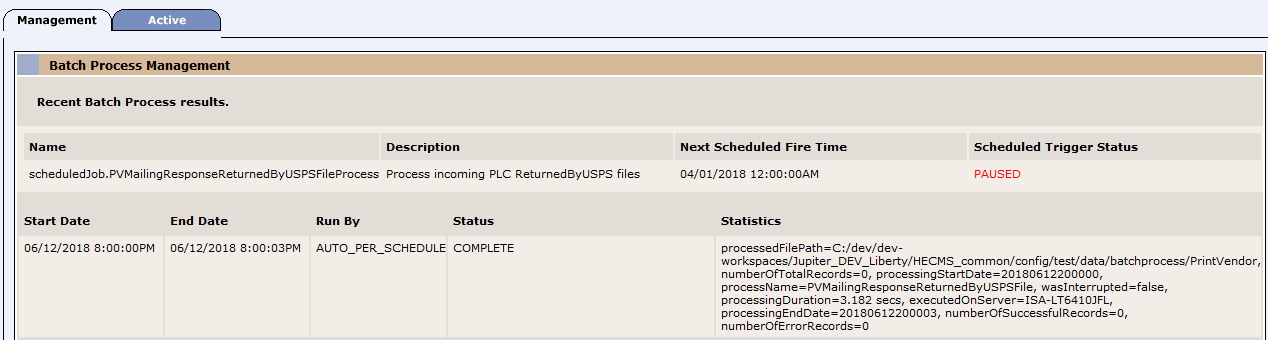


Figure 17: USPS Returned File Process Batch Process History View

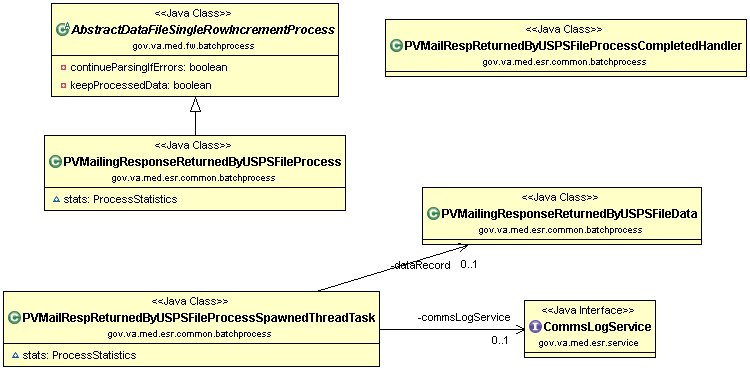


Figure 18: Class Diagram: USPS Returned File Batch Process

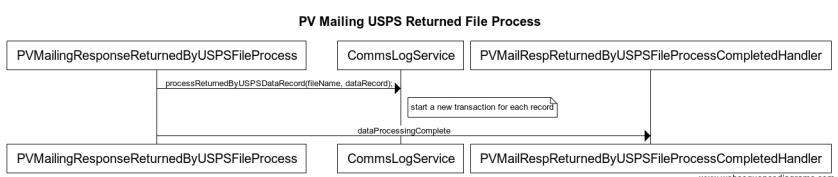


Figure 19: Sequence Diagram: USPS Returned File Batch Process

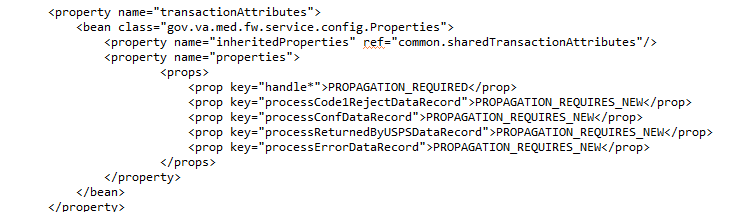


Figure 20: Spring configuration for transaction boundaries

## Pending Verification Mailing Error File for ScheduledJob.PVMailingResponseErrorFileProcess

Description: Batch Process will import CMS Errored mailing responses for the initial pending letters (742-650/742-651) and the new pre-closure letter (742-655).

**Main Class:** PVMailingResponseErrorFileProcess

**Extends:** AbstractDataFileSingleRowIncrementProcess (Throttles reading to 25 records each iteration)

**Transaction Boundaries.**

1. A new transaction will be started for each record by delegating the work to

commsLogService. processErrorDataRecord(dataRecord, fileName).

1. Process statistics will have their own transaction which are already configured as part of the framework.

**Exception Handling:** Any exception thrown out by the service will roll back the transaction, increment the error count and write the exception to the exception file. Exception file will have the same name as input file with a .exception extension.

**Multi-Threading:** This process run with seven threads with a task threshold of 10 tasks

**Input:**

1. **File naming pattern: PLC\_CMS\_error\_file\_\*.txt**
2. **Record Format:** formNum^uniqueId^vpid^element
3. **Sample Record:**

742-652A^40125502958^1008782311V208987^LETTERTHREE TESTERT

Error File Process Batch Process List View

Figure 21: Error File Process Batch Process List View

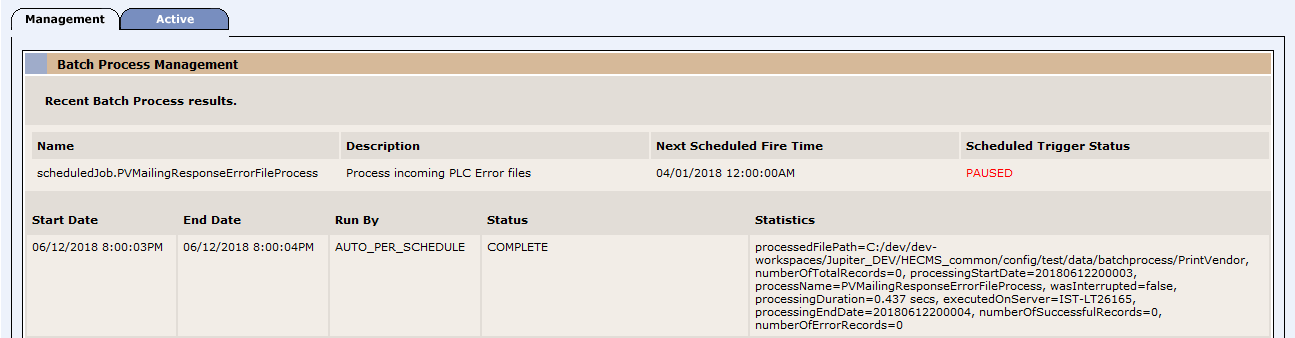


Figure 22: Error File Process Batch Process History View

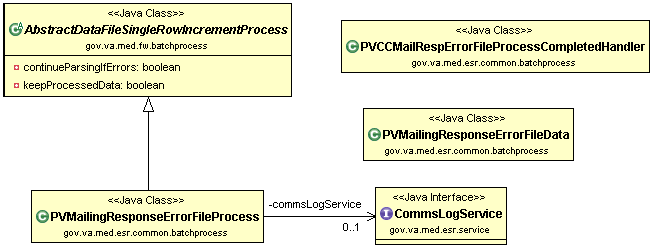


Figure 23: Class Diagram: Error File Batch Process

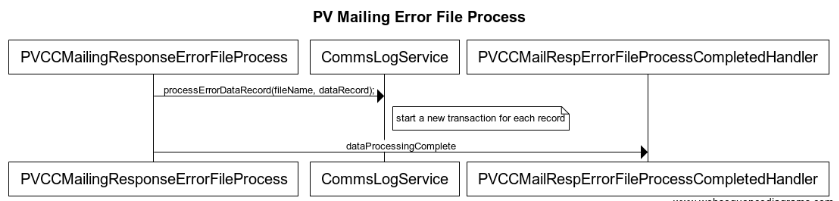


Figure 24: Sequence Diagram: Error File Batch Process

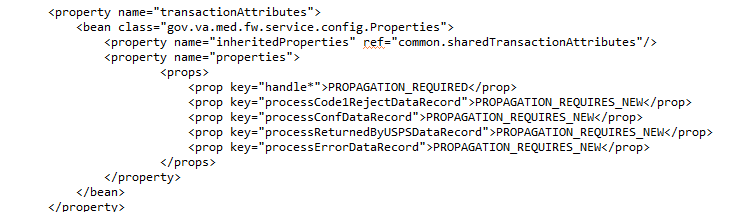


Figure 25: Spring configuration for transaction boundaries

## Activate VCE System Parameters

**Description**: This batch process implements RTC Story 734218: VCE Parameters: Batch Process to Activate System Parameters.

A future value and effective date have been added to the System Parameters table to support the VCE Parameters requirements. This batch process runs once a day and checks the effective date of each VCE parameter [ APP\_PARAMETER.PARAMETER\_EFFECTIVE\_DATE]. If the effective date is before or equal to the current day, the parameter is activated by:

* Copying the new value [ APP\_PARAMETER.PARAMETER\_NEW\_VALUE] to the current value [ APP\_PARAMETER.PARAMETER\_VALUE].
* Clearing out both APP\_PARAMETER.PARAMETER\_EFFECTIVE\_DATE and PARAMETER\_FUTURE\_VALUE.
* Sending an email to the configured email list.

**Main Class :** ActivateSystemParameters

**Extends :** AbstractDataProcess

**Associated Classes:** None needed, uses existing architecture.

**Transaction Boundaries.**

1. Default transaction boundaries are in effect, no new transactions are created. The System Parameter Service is used to read and update system parameters.
2. Process statistics will have their own transaction which are already configured as part of the framework.

**Exception Handling:** Any exception thrown out by the service will roll back the transaction, increment the error count and write the exception to the log file.

**Multi-Threading:**  No threads are created, it runs single threaded.

**Configuration:**

* Currently configured to run every day at 6 am in deployenv.properties

#### Activate System Parameters, Every 6:00 am Daily

batchProcess.ActivateSystemParameters.schedule=0 0 6 \* \* ?

* Email notification list is obtained from JOB\_CONFIG.EMAIL\_DISTRIBUTION\_LIST
* Spring configuration is in batch\_process.xml and scheduling.xml

**UI Support:** This batch process appears in the user interface (UI) under the description “Activate System Parameters”. From the UI you can execute it, update the email distribution list and review the history using the links provided on the **Batch Process Management** screen.

## VCE Parameters Changed Recalculation Batch Process

**Description**: These batch processes will recalculate Veterans Choice Eligibility for all active Veterans. [multiple instances exist]

**Main Class :** VCEParametersChangedProcess

**Extends :** AbstractDataQueryIncrementalProcess

**Associated Classes:** VCEParametersChangedSpawnedThreadTask

**Transaction Boundaries:**

* Default transaction boundaries are in effect, no new transactions are created. The System Parameter Service is used to read and update the system parameters.
* Process statistics will have their own transaction which are already configured as part of the framework.

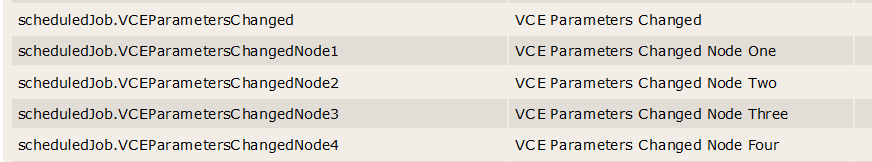
**Exception Handling:** Any exception thrown in the calculation process will increment the error count and the details of the exception will be written to the application log file.

**Multi-Threading:**  These processes are multi-threaded since it they will recalculate VC Eligibility on all active Veterans in the system. [ ~9 million ]

**Configuration:**

* They are not scheduled processes, they must be run manually through the Batch Process User Interface.
* They do accept custom arguments, discussed in the next section.

**Description:** In the UI, Enrollment System users will see:



There are 5 copies of this Batch Process in the UI. The implementation is identical for each but the data returned by the queries is different for each one.

**ScheduledJob.VCEParametersChanged:**

* This process has no restrictions on the data, it will do all veterans that need to be recalculated. [ VCELIGIBILITY. VCE\_RECALC\_REQUIRED\_IND is not set to ‘G’ ]
* The default number of threads is 20.
* The default Fetch Size is 10,000.

**ScheduledJob.VCEParametersChangedNode1-Node4:**

* These instances are restricted to ¼ of the total result set each. They are designed to run on different nodes in the production environment, for maximum throughput.
* The default number of threads is 15.
* The default Fetch Size is 10,000.

**Custom Arguments:**

Each process supports the same custom arguments, in this order:

1. Thread Pool size (integer)
2. Fetch Size (integer, number of records)
3. Job Result Update Interval (integer, number of records)
4. Task Count (integer)
5. Spawn Retry period (milliseconds)

The arguments can be separated by a comma or space, but not both.

**Process Logic:**

The processing of each record is as follows:

1. First the person is read from the db using the VPID returned by the query.
2. The VceParameterChangeInProcess flag is set on the Person.
3. The Person’s VcEligibility is read.
4. If a recalculation is necessary (VceRecalcRequired flag is not ‘G’) , then the eligibilityEnrollmentService.updateVCERecalculationForBatchProcess() method is called to invoke the business logic.
5. If VceRecalcRequired is set to ‘G’, the person is already recalculated and is skipped by the process.
6. The recalculation flag is set to ‘G’, to indicate that the recalculation was done.

**Thread Synchronization Points:** By design for record throughput, there are no thread synchronization points coded into these batch processes. The reason is that each process is designed to get different data. ScheduledJob.VCEParametersChangedwas left in the system to catch any possible records missed by the Node processes (for instance, if new records are created while the Node batch processes are running).

*Do not run* ***ScheduledJob.VCEParametersChanged*** *while the Node processes are running* because that may result in contention for the same data, and row locking db errors.